An Architecture for Enterprise Business Intelligence

A Review of the MicroStrategy Platform Architecture for Reporting, Analysis, and Monitoring Applications

MicroStrategy®
Best In Business Intelligence™

An Architecture for Enterprise Business Intelligence

A Review of the MicroStrategy Platform Architecture for Reporting, Analysis, and Monitoring Applications

MicroStrategy®
Best In Business Intelligence™

Table of Contents

INTRODUCTION **11**

1 Foreword **13**

2 Platform Design **17**
- 2.1 MicroStrategy's Philosophy – Delivering an Architecture for the Long Term 17
- 2.2 Architectural Tenets for Enterprise Business Intelligence 18
- 2.3 Complete BI Functionality Delivered on an Organically Developed Architecture 19
- 2.4 Enterprise BI on an Integrated Backplane and a Unified Web Interface 21
- 2.5 Summary 22

3 Product Architecture **25**
- 3.1 MicroStrategy has a Single Unified Architecture 25
- 3.2 MicroStrategy Intelligence Server 27
- 3.3 MicroStrategy Report Services 27
- 3.4 MicroStrategy OLAP Services 28
- 3.5 MicroStrategy Web and MicroStrategy Web Universal 28
- 3.6 MicroStrategy Narrowcast Server 28
- 3.7 MicroStrategy Office 29
- 3.8 MicroStrategy Desktop 29
- 3.9 MicroStrategy Architect 29
- 3.10 MicroStrategy BI Developer Kit 30
- 3.11 MicroStrategy Administrator 30
- 3.12 MicroStrategy SDK 31
- 3.13 MicroStrategy Transactor 31
- 3.14 Summary 31

PLATFORM COMPONENTS **33**

4 Metadata and the Logical Model **35**
- 4.1 A Complete Metadata is Key for Efficient Business Intelligence Processing 35
- 4.2 Configuration Objects 37
- 4.3 Data Abstraction Objects 38
- 4.4 Business Abstraction Objects 40
- 4.5 Report Components 42
- 4.6 Report and Document Definitions 43
- 4.7 Delivery Objects 44
- 4.8 Comprehensive Documentation of Metadata 45
- 4.9 Benefits of Metadata Design 46
- 4.10 Summary 50

5 MicroStrategy Intelligence Server Architecture 53
5.1 Powering the MicroStrategy BI Platform 54
5.2 Intelligence Server Engines and Query Flow 55
5.3 Automatic Resource Allocation 57
5.4 64-bit Business Intelligence 58
5.5 Optimized for Heterogeneous Data Sources 59
5.6 Sophisticated Analysis 63
5.7 Cluster-Capable Intelligence Server 68
5.8 Multi-Level Shared Caching 69
5.9 Scheduling Reports and Administrative Tasks 73
5.10 Exporting Data 74
5.11 Formatting Data 75
5.12 Simplified Central Administration 75
5.13 Summary 79

6 MicroStrategy Web Architecture 81
6.1 Core Design Principles 81
6.2 Optimize Performance with a Service-Oriented Architecture 81
6.3 Open, Layered Design Optimizes Performance, Integration, and Customization 83
6.4 A Single, Platform-Independent Code Base Provides Flexibility and Consistency 86
6.5 All Data in the BI Application is Secure 87
6.6 Employ Leading Edge Technologies to Maximize Ease-of-Use and Interactivity 88
6.7 Summary 90

7 MicroStrategy Narrowcast Server Architecture 93
7.1 Narrowcast Server is the Report Delivery Engine 93
7.2 Architecture and Workflow 95
7.3 Creating Report Delivery Services 99
7.4 Deploying Non-Interactive Content using the Subscription Portal 101
7.5 Architected For Enterprise-class Report Delivery 102

PLATFORM FUNCTIONALITY 105

8 Security 107
8.1 Securing Business Intelligence Applications 107
8.2 User Authentication 108
8.3 User Authorization 110
8.4 Multi-Tier Web Architecture and Transmission Security 116
8.5 Enterprise-Class Security at Every Level of the BI Architecture 118

9 Reliability and Fault Tolerance 121
9.1 Stateful and Stateless Architectures 121
9.2 Modular Components Can Be Kept Together or Separated 123
9.3 Shared, Unified Metadata 123
9.4 Communication Between Cluster Nodes 123
9.5 MicroStrategy Clustering Capabilities 124
9.6 Automatic Failover 127
9.7 Clustering for Batch-Processing and Report Distribution 128
9.8 Summary 129

10 Scalability and Performance 131
10.1 Query Optimization 132
10.2 Multi-Level Shared Caching 139
10.3 Distributable Execution 142
10.4 64-bit Business Intelligence 145
10.5 Multi-Threaded Processing 146
10.6 Efficient Communications 148
10.7 Proven Scalability and Performance 150

11 Exporting Data and Reports **153**
11.1 MicroStrategy Exports Enhanced Data 153
11.2 MicroStrategy Provides Efficient Exporting 154
11.3 MicroStrategy Office Integrates With Excel, Word, and PowerPoint 158
11.4 MicroStrategy Supports Flexible Exporting for All Uses 160

12 Data Mining and Predictive Analysis **163**
12.1 Integrating Data Mining Into Business Intelligence 164
12.2 Data Mining Services in MicroStrategy 166
12.3 Data Mining Algorithms 167
12.4 Benefits of Integrating Business Intelligence and Data Mining 172

13 Closed-Loop Business Intelligence **175**
13.1 Closed-Loop Applications Reduce the Time to Act 175
13.2 Closed-Loop BI Using Dynamic Datamarts 176
13.3 Closed-Loop BI Using Freeform SQL 177
13.4 Closed-Loop BI Using MicroStrategy Transactor 178
13.5 Summary 184

USER INTERACTIVITY **187**

14 End User Experience **189**
14.1 Consistent Reports, Functionality, and Security through Many User Interfaces 189
14.2 The Business Intelligence Application Spectrum 193
14.3 MicroStrategy User Experience Design Tenets 194
14.4 Scorecards and Dashboards 195
14.5 Reporting 198
14.6 OLAP Analysis 201
14.7 Advanced and Predictive Analysis 206
14.8 Alerts and Proactive Notifications 208
14.9 User Functionality by User Role 208
14.10 Customizing MicroStrategy Functionality for the End User 212
14.11 Summary 213

15 Developer Experience **215**
15.1 Multiple Interfaces for BI Development 215
15.2 The Business Intelligence Application Spectrum 215
15.3 Map Database Structures to Qualitative and Quantitative Business Terms 219
15.4 MicroStrategy Desktop 222
15.5 Business Abstraction Objects 224
15.6 Report Objects 228
15.7 Report Creation Using the Dynamic SQL Engine 228
15.8 Creating OLAP Services Cube Reports 230
15.9 Build and Execute Reports with the Dynamic MDX Engine 231
15.10 Operational Reports Using the Freeform SQL Engine 232
15.11 Building Graphs in MicroStrategy Desktop 233
15.12 Data Mining and Predictive Reports in MicroStrategy Desktop 234
15.13 Report Services Document Creation 235
15.14 Web Professional 238
15.15 Creating Pixel-Perfect Formatted Dashboards and Scorecards over the Web 242
15.16 WYSIWYG Editing of Existing Reports and Documents Over the Web 243
15.17 Narrowcast Server 244
15.18 Narrowcast Server Design Process 245
15.19 Customization / SDK Developer 248
15.20 Summary 250

IMPLEMENTING BI APPLICATIONS **269**

16 Administrator Experience **253**
16.1 In-depth Administration across the Whole Platform 253
16.2 Real-Time Monitoring and Control 254
16.3 Error Logging and Diagnostic Files 256
16.4 Enterprise Manager – Historical Performance Analysis 257
16.5 Centralized Administration 258
16.6 System Management Software Integration 265
16.7 Summary 266

17 Developing Business Intelligence Applications **271**
17.1 Application Lifecycle 271
17.2 Evolve from Departmental to Enterprise BI Applications 274
17.3 Estimating the Capacity Requirements of the BI Application 279
17.4 Summary 283

18 Portable Analytic Modules **285**
18.1 Rapid Application Development Framework 285
18.2 Portable Analytic Applications 286
18.3 MicroStrategy BI Developer Kit Analytic Modules 287
18.4 Summary 297

19 Extensibility and the SDK **299**
19.1 Access All BI Functionality in a Service-Oriented Architecture with Open APIs 299
19.2 MicroStrategy Web API 300
19.3 MicroStrategy Intelligence Server API 304
19.4 MicroStrategy Narrowcast Server API 304
19.5 MicroStrategy Office API 305
19.6 MicroStrategy Transactor API 305
19.7 Comprehensive Documentation that Describes the API in Detail 305
19.8 Simplify Development through Utilities and Sample Code 306
19.9 Portal Integration 308
19.10 Web Services Development 309
19.11 Sample Applications 309
19.12 Customization and Extension Examples 311
19.13 Summary 317

APPENDICES **319**

Appendices **321**
Appendix A: Schema Support 321
Appendix B: Very Large Database (VLDB) Properties 324
Appendix C: Analytical Functions 328
Appendix D: Supported Graph Types 330
Appendix E: Performance Counters and Key Performance Indicators 331
Appendix F: Supported Data Sources 334
Appendix G: Security Privileges 336
Appendix H: API Classes, Methods, Properties, and Interfaces 337

INTRODUCTION

1

Foreword

Partnering With Our Customers to Achieve Enterprise Business Intelligence

The MicroStrategy® architecture that you see today is the culmination of roughly 3,000 man-years of engineering effort and well over half a billion dollars in engineering investment. More significant is the fact that the MicroStrategy architecture embodies the cumulative engineering advances gained through over 15 years of experience in supporting the most demanding BI applications in the world.

Our customers drive our technology development. Ever since MicroStrategy's first customer implementation, we have been challenged to solve the most technically difficult problems in BI. After countless implementations for over 3,000 customers, we fully understand the value of a close partnership of co-development and BI innovation that is continually fueled by our customers' uniquely aggressive and ever-expanding requirements. The effort we have invested in supporting our customers' initiatives and applications has propelled our software faster and farther than any competing BI technology on the market.

Very early we realized that our customers' BI requirements would continue to grow as their data volumes and user populations increased, and as their business requirements became more broader and sophisticated. In fact, it became clear to us that our customers were driving the technology toward the goal of "Enterprise-wide Business Intelligence." We took this goal to heart when we designed our new product architecture – beginning with MicroStrategy 7™ in 2000, and evolving to MicroStrategy 8™ by 2005.

Nearly every BI vendor now claims to have an "integrated BI architecture" suitable for Enterprise BI standardization. What is often overlooked is the fact that MicroStrategy's end goal was not integration – it was the starting point. We view integration not as a feature but as a requirement for a scalable, maintainable, and high performance architecture. Any architecture with duplicative or redundant metadata, data structures, processes, queues, caches, and thread management facilities will possess inherent performance inefficiencies, be difficult to manage, and be more costly to maintain. Given that performance and scalability have always been the dominant limiting factors to successfully deploy enterprise systems, an architecture designed from the start for Enterprise BI was needed – not one evolved from a set of existing products and legacy code.

As such, we determined that we needed to discard every line of code in our previous architecture (version 6 and prior) and start over from scratch. While this business decision was considered risky at the time, the strategy was necessary if we wanted to meet the long-term needs of our customers. It allowed us to start by designing an architecture built specifically for performance and maintainability. Additionally, it allowed us to build new features as part of an integrated architecture, rather than worry about integrating legacy products after the fact. Interestingly, none of the competitive BI technologies followed our approach. Instead, they decided to make incremental gains towards integration, and chose architectures built primarily to achieve legacy integration rather than for performance and flexibility. In retrospect, our strategy has turned out to have extraordinarily beneficial consequences for both MicroStrategy and our customers.

The result of our efforts is MicroStrategy 8, which is presented in this book. Our architecture can be compared to an iceberg where 90% of the mass lies under the surface, hidden from casual view. Often, our customers do not begin to appreciate the full depth and thoughtfulness of our product's capabilities until they have used the technology for several years, and grown their BI implementation into a world-class application beyond their initial expectations and vision.

We offer this book as a means of formally presenting our architecture – coded by our engineers and co-developed by our customers for over 15 years. I hope the content will help provide a level of detail that is lacking in most BI literature today, and provide some clarity as to what you can achieve with the MicroStrategy Platform. We believe that we have a unique offering and are confident that as you dig into the detail, you will conclude the same.



Jeffrey A. Bedell
Chief Technology Officer, MicroStrategy

2

Platform Design

Since 1989, MicroStrategy's engineers and designers have methodically tackled, and solved the hardest technical problems in business intelligence – delivering the greatest user interactivity, against the largest databases, with the most sophisticated analytics, and accessed by the largest number of users. Today, MicroStrategy technology supports the most demanding BI applications across all industries.

2.1 MicroStrategy's Philosophy – Delivering an Architecture for the Long Term

MicroStrategy was designed specifically to usher in a new era in business intelligence, with an architecture that is organically integrated and expressly designed for Enterprise BI. All components of MicroStrategy's architecture are the result of organic growth from a single architectural baseline. The "integrated BI architectures" from other vendors are not organic, but are the result of integrating proprietary BI technologies, some of which are more than 10 years old.

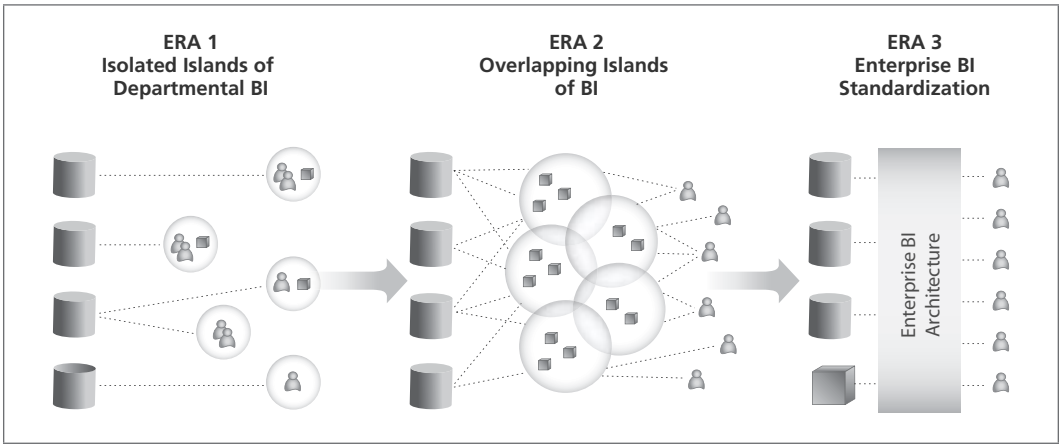


Figure 2.1 MicroStrategy's organically integrated architecture was specifically designed to usher in a new era of enterprise BI, and help enterprises evolve from previous eras of islands of disparate BI technologies and inconsistent departmental BI applications.

1. Era 1: Isolated Departmental Islands of BI Are an Initial Success
   Throughout the 1990s, most companies deployed BI applications as departmental solutions, and accumulated a large collection of disparate BI technologies as a result. Each distinct technology supported a specific user population and database, within a well-defined "island of BI." At first, these islands of BI satisfied the initial needs of the business, but early success in departmental deployment sowed the seeds for new problems as the applications grew.

2. Era 2: Overlapping Disparate Islands of BI Have Become an Enterprise Liability
   Successful applications always expand. The second era of BI is hallmarked by BI applications that have data

access, and analytic coverage. As a result, CIO's are now faced with an untenable situation. The enterprise is getting conflicting versions of the truth through the multiple disparate BI systems, and there is no way to harmonize them without an extraordinary ongoing manual effort of synchronization. Equally problematic is the fact that business users are forced to use many different BI tools depending on the data they want.

3. Era 3: Enterprise BI Standardization Delivers a Single Version of the Truth with Lowest Total Cost of Ownership
   The new era of business intelligence is one where a single BI architecture delivers one version of the truth through a single user interface to all people across the enterprise. It can access all of the data, administer all of the people, eliminate repetitive data access, reduce the administrative effort, and reduce the time to deploy new BI applications. This is not just a vision of Enterprise Business Intelligence. Many companies are successfully achieving this new era of business intelligence with MicroStrategy technology.

The bedrock of MicroStrategy's architectural philosophy is to deliver a living architecture – one that can continually evolve to greater functionality, greater scalability, with seamless integration, and ever greater economies of scale. Adhering to this basic philosophy often entails making difficult technical choices where the tradeoff is between expediency (building something quickly just to satisfy the immediate market requirement) vs. long-term flexibility (designing something with primary emphasis on extensibility and maintainability over time).

Unlike software companies that are driven by the expediency of day-to-day market trends, MicroStrategy has never wavered from its commitment to building a long-term enterprise-caliber technology. In the end, the market will decide whether it is satisfied with casual BI architectures suitable for standalone departmental BI applications, or whether it wants industrial-strength BI technology that can support many BI applications at enterprise scale. MicroStrategy firmly believes that companies ultimately want enterprise BI, and have based over 15 years of architectural decisions on that overarching belief.

Throughout this book there are many examples of where MicroStrategy seemingly chose the most technically difficult route. The reason is that our designs invariably address far more rigorous requirements than are immediately apparent. In fact, every design decision we take must explicitly adhere to seven architectural tenets for long-term flexibility and Enterprise BI usage.

2.2 Architectural Tenets for Enterprise Business Intelligence

1. Scalability and High Performance
   All design decisions must ensure that MicroStrategy can deliver consistently high performance as the system scales upward, and must anticipate order-of-magnitude growth beyond today's state-of-the-art standard (user scale, data scale, and application scale).

2. Economies of Scale
   All design decisions must explicitly deliver greater economies of scale as a system grows – using techniques like caching, object reuse, automated administration, and collaborative analytics. We make sure that MicroStrategy implementations require an absolute minimum of IT personnel, require the fewest servers, and minimize the workload on expensive database resources.

3. Complete Functionality
   All design decisions must ensure that the MicroStrategy architecture offers the full range of BI functionality on a single services-oriented architecture, so that customers can satisfy all of their BI requirements without the need for additional integration work.

4. Incremental Growth
   All design decisions must ensure that customers can incrementally grow their BI infrastructures – from small to large, from departmental scope to enterprise scope, from isolated islands to consolidated applications, and from reporting to dashboards to OLAP to ad hoc analysis to alerting. This allows our customers to initially buy just the functionality they need, and to incrementally grow their BI solution as their requirements naturally expand. We have designed our plug-n-play components so that each added component brings new functionality to all pre-existing components and builds on pre-existing content so that no rework is required.

5. Openness and Extensibility
   All design decisions must ensure that MicroStrategy's vast functionality continues to be fully accessible through Web services APIs. The degree of commitment MicroStrategy has toward its APIs is best illustrated by the fact that MicroStrategy technology internally uses the same thousands of APIs that we expose externally in our SDK documentation.

6. Centralized Consistency with Distributed Governance and Self-service
   All design decisions must support the goal of a consistent single version of the truth throughout the enterprise using a single shared metadata and a pervasive security architecture. Yet, the architecture must also provide a high degree of autonomy to distributed development teams (managed by departments and divisions), and to individual users allowing them to create enterprise-consistent solutions at a local level.

7. Rapid Development and Deployment
   All design decisions must promote rapid development and deployment of new reports and applications. MicroStrategy has invested significant engineering energy in creating a vast array of reusable metadata objects, by creating a security architecture that is applied automatically, and pervasively with no effort on the part of the report designer, and through WYSIWYG design paradigms that allow novices and experts to play a role in accelerating the report design process.

These requirements do not come from MicroStrategy, they are requirements from our long-time customers who originally invested in MicroStrategy technology to support their high-scale BI applications. Today, the goal of these same customers is to host many diverse BI applications on the same consistent Enterprise BI architecture and the architectural tenets ensure their long-term success in this process.

2.3 Complete BI Functionality Delivered on an Organically Developed Architecture

The 1990s represented a formative period for business intelligence applications and products. During this period, organizations realized that they needed a range of BI capabilities to satisfy a diverse set of user needs. Ultimately, this range could be grouped into five distinct categories or "Styles of BI" applications. The 5 Styles of BI applications include:

1. Scorecards and Dashboards
2. Reporting
3. OLAP analysis
4. Advanced and Ad hoc analysis
5. Alerts and Proactive Notification

In response to this fragmented market demand, most BI vendors developed different products, each designed to support one of the 5 Styles of Business Intelligence.

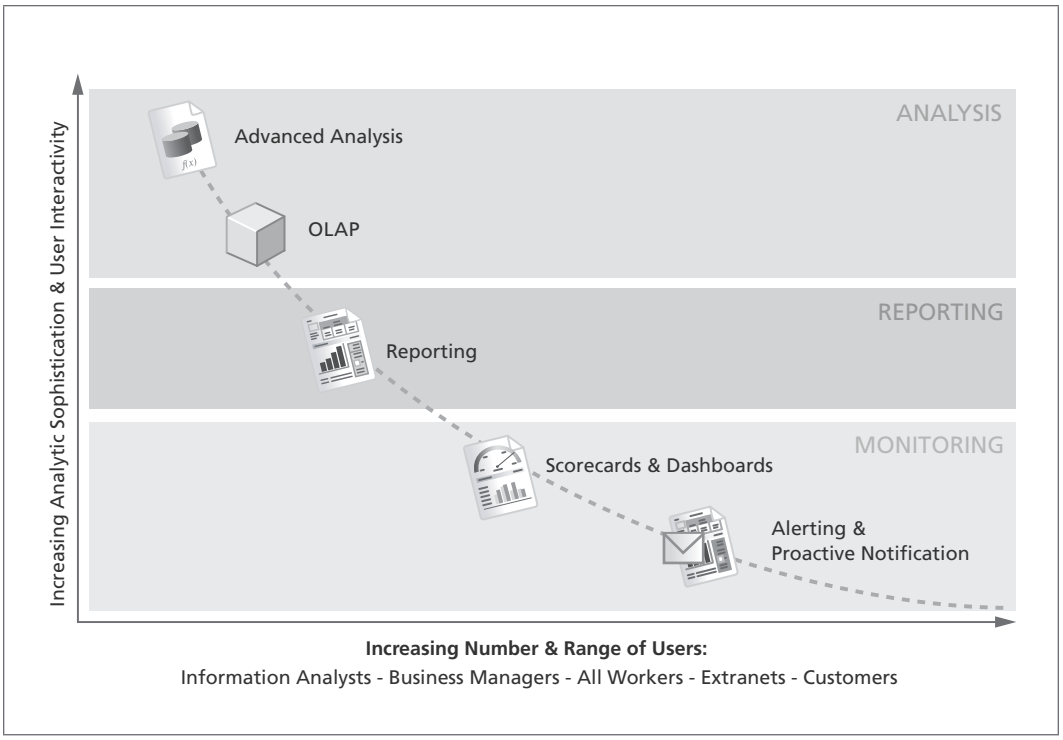


Figure 2.2 The MicroStrategy architecture offers all 5 Styles of business intelligence with a single organically-developed architecture.

Companies adopted a variety of BI products, and built BI applications where each application was centered on a single style of business intelligence. Today, companies have concluded that the boundaries between Styles of BI (e.g., the boundaries between reporting and scorecards and analysis) are not natural business boundaries, but are artifacts of BI product limitations. In the ideal world, all BI products would be able to seamlessly support any or all 5 Styles of Business Intelligence.

Microstrategy Delivers All 5 Styles of Business Intelligence

Beginning in 1996, MicroStrategy embarked on a massive software development effort to re-engineer its highly scalable architecture so that it could support all 5 Styles of Business Intelligence. By 2000, MicroStrategy delivered MicroStrategy 7 which laid the foundation for this vision, and explicitly supported the two most difficult styles of BI: 1) Advanced and Ad hoc Analysis, and 2) Alerting and Proactive Notification. By mid-2002, MicroStrategy added the OLAP Cube style of BI. In 2003, MicroStrategy completed the functionality set by adding the Reporting and Scorecard & Dashboard style of BI.

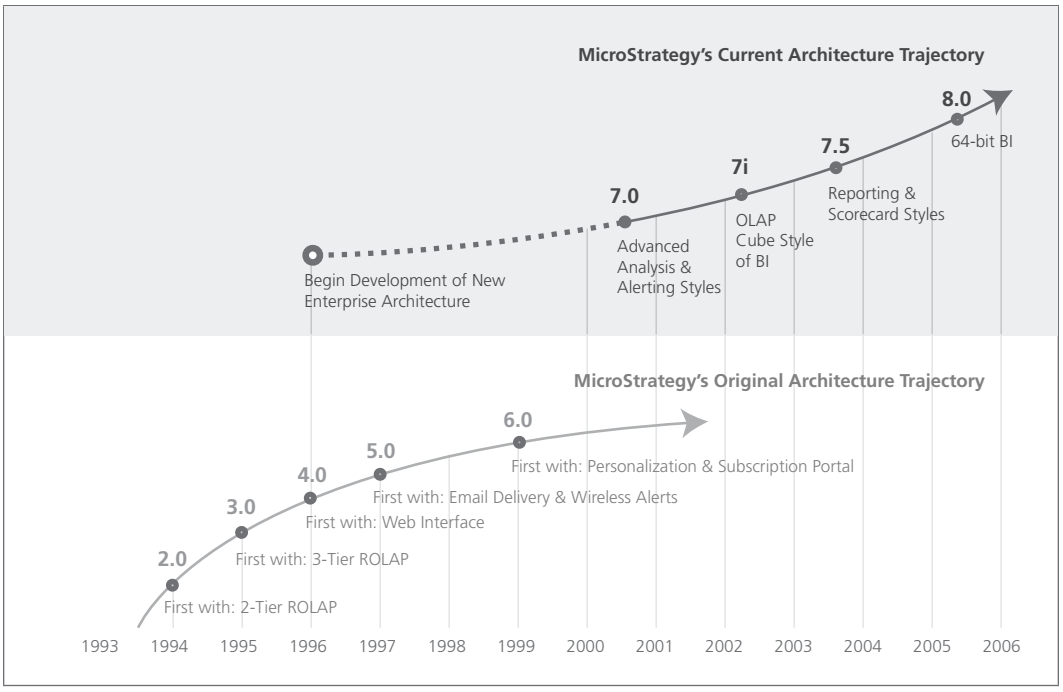


Figure 2.3 Despite a series of market innovations on MicroStrategy's original software architecture, MicroStrategy embarked on a total reengineering of the software in 1996 to produce an architecture capable of supporting all 5 styles of business intelligence.

2.4 Enterprise BI on an Integrated Backplane and a Unified Web Interface

The modern MicroStrategy architecture delivers all 5 Styles of business intelligence on a unified backplane of common services and through a single Web architecture.

Service Oriented Integrated Backplane

The integrated backplane is the heart of the MicroStrategy architecture. It provides the common services of metadata, prompt generation, scheduling, shared caching, security, user management, query generation, query governing, and administration. More importantly, it is the core engine which supports each of the 5 Styles of BI as plug-n-play "service modules" that can be mixed and matched in any combination. These modules include MicroStrategy Report Services for pixel perfect reporting, dashboards and scorecards, MicroStrategy OLAP Services for speed-of-thought slicing and dicing, and MicroStrategy Narrowcast Server for alerting and proactive notification functionality. MicroStrategy ROLAP and analytics are always an integral part of the core engine.

MicroStrategy customers can add various "service modules" to the core engine to incrementally extend its functionality. Each service module is designed to build on one another, adding new functionality to each other when used in combination. This means that a customer can add the OLAP Services module to the engine, and all previously-built grid reports automatically become Intelligent Cube™ reports, and inherit a wide range of new functionality. In addition, a customer can add the Report Services module, and be able to re-use all the previously-built grid and cube reports as datasets for the new pixel-perfect document. And finally, it means that a customer can add the Narrowcast Server module, and immediately use any grid reports, Intelligent Cube reports, and document reports as the basis for email alerts and proactive notifications.

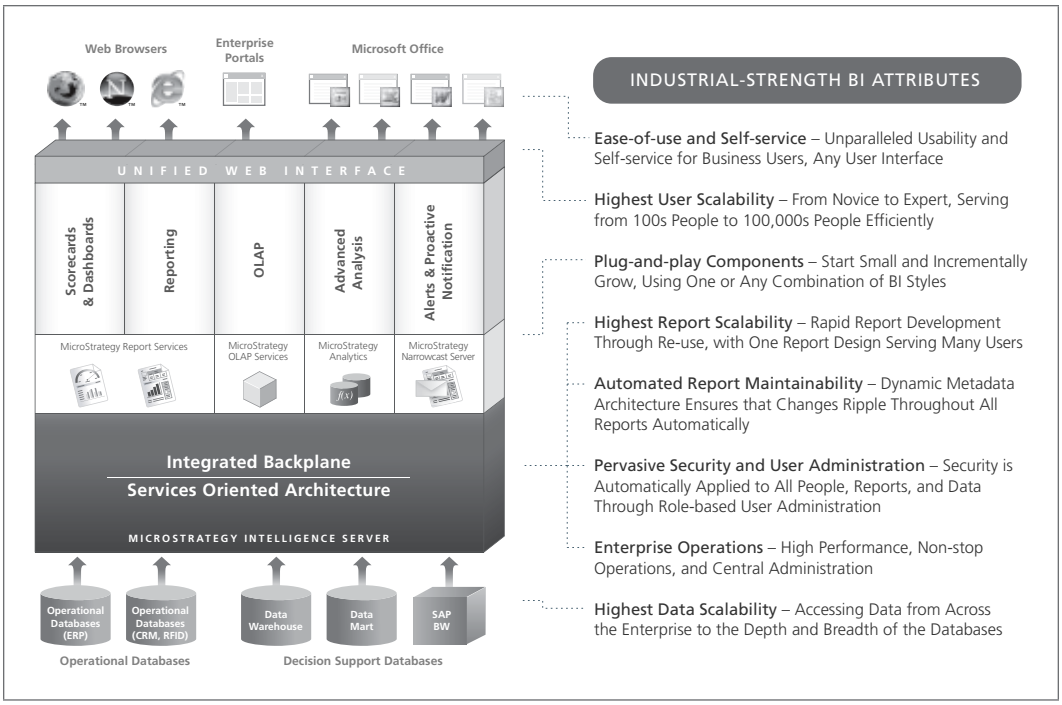


Figure 2.4 MicroStrategy is a uniquely unified architecture – delivering all 5 styles of BI through a unified user interface and on a unified backplane of metadata and administration.

MicroStrategy's Unified Web Interface

MicroStrategy's unified Web interface is a single code base that provides BI functionality to all users, ranging from basic reporting functionality for novice users, to interactive analysis functionality for analysts, to full report design functionality for power users and IT professionals. Because of the way it is designed, the MicroStrategy Web interface can deliver BI functionality equally well to multiple different interfaces, including: different Web browsers, third-party enterprise portals, and even to the Microsoft® Office applications of Excel®, Word, PowerPoint®, and Outlook® using Web services.

From a user point of view, MicroStrategy's Web interface is a truly unified interface. It presents the same screen layout, the same drag-n-drop actions, the same menus, and the same toolbars regardless of user skill level and regardless of which style of BI a person is using. By contrast, competing "integrated Web interfaces" from other vendors are hardly unified at all. They present different screen layouts, different drag-n-drop actions, and different drop-down menus for each of their various "integrated interfaces" because many of those interfaces were acquired or were developed independently of one another over an extended period of time.

2.5 Summary

MicroStrategy uniquely offers the only enterprise-class business intelligence technology suitable for the most demanding BI applications, suitable to deliver all 5 Styles of BI seamlessly, suitable to be the BI standard across the entire enterprise. The intent of this book is to justify this statement. To do so, this book will discuss in detail all aspects of MicroStrategy's elegant platform architecture. Organizationally, this book approaches the MicroStrategy platform as one might approach beginning development of a business intelligence application – starting at the infrastructural components, and working out to a finished application in production.

3

Product Architecture

The MicroStrategy business intelligence platform is the culmination of over 15 years of BI experience. MicroStrategy offers the only enterprise-class BI technology suitable for the most demanding BI applications, suitable to deliver all 5 Styles of BI seamlessly, and suitable to be the BI standard across the entire enterprise. This enterprise-class technology ensures:

- Complete range of business user functionality – organically integrated scorecards, dashboards, reporting, analysis, alerts, and proactive notification.
- Depth of analytical power – full investigative analysis of the entire data warehouse, not just subsets of data, including data mining, predictive, statistical, financial, and mathematical analysis.
- Comprehensive administrative control – consistent with mission-critical system operation.
- Limitless application extensibility – open APIs exposed through industry standard Web services, Java™, XML/XSL, xHTML, AJAX, CSS, and COM to integrate and customize applications with other mission-critical systems.

3.1 MicroStrategy has a Single Unified Architecture

The MicroStrategy platform is built on one, unified, organically-developed architecture. To achieve the seven design tenets for Enterprise BI, the underlying architectural design establishes:

1. A single, unified object model to define and construct objects that represent any business
2. An organic, platform of 11 distinct products that dynamically assemble, and re-assemble these objects in response to any business question or user action.

All platform products use object-oriented programming techniques of inheritance and encapsulation ensuring scalability and high performance.

The MicroStrategy object model is the genius of the MicroStrategy platform. The MicroStrategy metadata is the manifestation of the object model. The metadata contains the building blocks or objects necessary to represent an enterprise's business. The metadata stores these objects in a database for efficient re-use, manageability, and performance. The objects themselves are used to create new objects such as a report definition. Hence, if one object changes, every other object dependent on it also changes. This ensures consistency across business definitions, and actually minimizes the number of objects created, stored, and managed. What truly makes the MicroStrategy metadata unique is that it only stores these objects; it does not store a finished report complete with a static SQL statement. Because of this, the MicroStrategy metadata gives the MicroStrategy platform unparalleled analytical, functional, and administrative flexibility.

The MicroStrategy BI platform dynamically assembles the metadata objects to create reports, scorecards, dashboards, analyses, and alerts. The platform provides all 5 Styles of BI uniformly through a wide variety of user interfaces, including Microsoft Windows® workstations, Enterprise Portals, Microsoft Office, Web browsers, email, network printers, file servers, wireless PDAs and mobile phones. The MicroStrategy platform comprises of tightly integrated products that deliver unmatched functionality, power, control, and extensibility on enterprise-class technology.

Architectural Functionality	MicroStrategy Products
Serving Reports, Scorecards, Dashboards, OLAP, Advanced, and Ad Hoc Analysis, and Alerts and Proactive Notification	Intelligence Server, Web, Narrowcast Server, Report Services, OLAP Services
Business User Interactivity	MicroStrategy Web, MicroStrategy Office, Report Services, OLAP Services, Desktop
Batch Reporting	Narrowcast Server, Intelligence Server, Report Services, OLAP Services
Application Development (Metadata Object creation)	Architect, Desktop, Web
Administration	Administrator , Intelligence Server
Integration With Other Applications	SDK, MicroStrategy Office, Transactor

Figure 3.1 The MicroStrategy products associated with the architectural functionality they provide.

Every MicroStrategy BI application requires at a minimum:

1. Development products to build business abstraction objects in the metadata: MicroStrategy Architect and Desktop.
2. Core server to manage users, security, database communications, and report formatting and distribution: MicroStrategy Intelligence Server.
3. Interface products to run reports and manipulate the results for investigative analysis: MicroStrategy Web, MicroStrategy Office, or Narrowcast Server

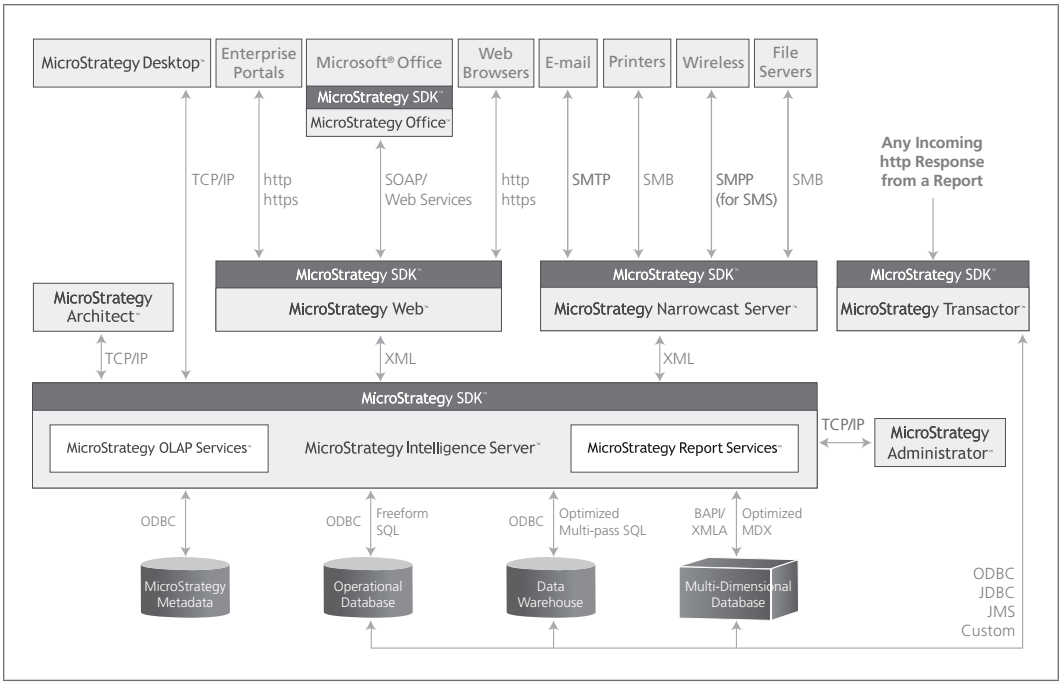


Figure 3.2 The MicroStrategy product architecture provides the full range of functionality, power, control, and extensibility on the industry's only enterprise-class BI technology. All business user interactivity is uniformly expressed through a Windows workstation, enterprise portals, Microsoft Office, Web browsers, email, network printers, file servers, or wireless devices.

Depending on the application demands, any number of products can be used together to scale to large business user populations. For example, MicroStrategy Narrowcast Server can distribute reports directly to business users via email who simply need static reports and not the interactivity of MicroStrategy Web. MicroStrategy Office is particularly popular with finance departments who like to do all their work in Microsoft Excel.

Beyond these products, the MicroStrategy SDK is ideal to deliver MicroStrategy reports directly to enterprise portals. Certainly mid to large BI environments would benefit from MicroStrategy Administrator to centrally monitor, and manage the entire business intelligence infrastructure. If the BI applications requires sophisticated closed-loop, actionable BI reports, MicroStrategy Transactor lets business users initiate transactions with backend systems and write back to databases. The MicroStrategy BI Developer Kit offers six analytical BI applications to get started quickly. The products that comprise the MicroStrategy platform are briefly described next.

3.2 MicroStrategy Intelligence Server

MicroStrategy Intelligence Server™ is the architectural foundation of the MicroStrategy platform. As the central contact point to the metadata, Intelligence Server dynamically assembles the metadata objects to create optimized, multi-pass SQL queries for every major relational database. Intelligence Server retrieves the data, performs any additional analytical calculations not available in the databases, formats the report, and delivers the reports to business users via MicroStrategy Web, MicroStrategy Office, Desktop, or Narrowcast Server.

Intelligence Server is a highly scalable, parallel-processing, self-tuning analytic server. Intelligence Server manages high performance interactions accessing terabytes of data by tens of thousands of users using caching, load balancing, resource prioritization, and connection pooling. It accesses and joins data from multiple data sources, such as data warehouses, operational databases, multi-dimensional (cube) databases, and even flat files. Intelligence Server also manages users, system security, data security, and user functionality access. A clustering option is available with Intelligence Server that increases scalability, and provides fault tolerance with automatic failover.

3.3 MicroStrategy Report Services

MicroStrategy Report Services™ is a plug-n-play extension to Intelligence Server providing pixel-perfect, print-perfect, and page-perfect report design and distribution through MicroStrategy Web, MicroStrategy Office, Desktop, and Narrowcast Server. Report Services delivers the most exacting report formatting for any of the following:

- Operational Reports – classic production reporting requiring print-perfect layout with data organized and aggregated into hierarchies or bands of increasingly finer detail
- Scorecards and Dashboards – highly visual, pixel-perfect displays that provide "at-a-glance" view of the enterprise using gauges, dials, KPIs, and metric status
- Invoices and Statements – page-perfect layouts designed for billing applications and statutory reporting
- Business Reports – any report format, usually combining graphs, detail data, and often explanatory text, used to describe business performance

Report Services reports, scorecards, dashboards, invoices, statements, and business reports have fully interactive analytical and WYSIWYG design capabilities over a zero-footprint Web browser.

3.4 MicroStrategy OLAP Services

MicroStrategy OLAP Services™ is a plug-n-play extension to Intelligence Server that adds classic multi-dimensional or OLAP cube functionality to the standard ROLAP functionality of the MicroStrategy BI platform. OLAP Services creates and distributes Intelligent Cubes, a multi-dimensional-cache structure beneath a MicroStrategy report. OLAP Services exposes the report objects (metrics and attributes) in the Intelligent Cube so that analysts can manipulate report objects, create derived metrics, and modify filter criteria. Because all of these analytical actions take place on the Intelligence Server rather than the database, OLAP Services delivers speed-of-thought analysis.

One frequent user action is to drill down into the data. With OLAP Services, the Intelligent Cubes can contain attributes commonly requested via drilling. This greatly enhances the report performance because OLAP Services can return results without querying the data warehouse. OLAP Services allows seamless drill through to the data warehouse to investigate the full breadth and depth of the corporate data assets.

3.5 MicroStrategy Web and MicroStrategy Web Universal

MicroStrategy Web™ and MicroStrategy Web Universal™ provide a powerful and user-friendly environment for interactive analysis through any Web browser. Suitable for all levels of user sophistication from beginner to advanced, MicroStrategy Web and Web Universal generate an interactive BI experience including report viewing, formatting, exporting, pivoting, sorting, drilling, and ad hoc querying to WYSIWYG report design and creation. Using advanced Web technology including xHTML, CSS, and AJAX, MicroStrategy Web and Web Universal deliver a Windows-over-Web user experience spanning the full range of MicroStrategy functionality.

Both MicroStrategy Web and Web Universal accomplish all of its functionality through a cookie-less, zero-footprint Web client without using ActiveX® or Java Applets resident in or downloaded to the Web browser. Thus, MicroStrategy Web and Web Universal work seamlessly through all standard Web firewalls making deployment straightforward and minimizing administration overhead.

MicroStrategy Web provides the Web interface using Active Server Pages (ASP) on 32-bit processor computers running Microsoft Internet Information Service (IIS) on the Microsoft Windows® operating system. In contrast to MicroStrategy Web, MicroStrategy Web Universal provides the identical end-user functionality through ASP or Java Server Pages (JSP) on either 32- bit or 64-bit processor computers running any supported application servers on any operating system.

Note that throughout this book, all references to MicroStrategy Web include both Web and Web Universal.

3.6 MicroStrategy Narrowcast Server

MicroStrategy Narrowcast Server™ proactively distributes personalized reports and alerts. In contrast to MicroStrategy Web's interactive BI environment, Narrowcast Server delivers static reports. Often, the largest number of users simply needs to receive static reports. Those reports, however, must come automatically based on one or more of the following:

- A time schedule
- An event such as a completed database load
- A trigger such as business metric tracking outside an acceptable range

Serving these large user communities, Narrowcast Server can deliver over 100,000 reports per hour. Because it is critical these users receive this information without having to request it, Narrowcast Server automatically sends those reports to a variety of devices, including email, file servers, networked printers, and wireless devices such as mobile phones and PDAs. Narrowcast Server also includes a Web-based Subscription Portal for users to self-subscribe to and receive reports, input personalized criteria, and set scheduling, event, or data-trigger criteria. Business users can also subscribe to receive Narrowcast Server's proactive notification and alerts from MicroStrategy Web. In both cases, this significantly reduces administrative burden, especially when the users number in the tens of thousands.

The proactive notifications and alerts Narrowcast Server sends include the precise report formatting for the desired output device. The report can also include multiple attachments including:

- Report Services enterprise documents or scorecards in PDF format
- MicroStrategy reports in Microsoft Excel workbooks
- Large datasets in CSV format
- Zip files containing any of the above

3.7 MicroStrategy Office

MicroStrategy Office™ delivers MicroStrategy reporting and analysis to Microsoft Excel, PowerPoint, Word, and Outlook using Web services. This is ideal for business users who want to use Microsoft Office for their BI interface. Excel, PowerPoint, and Word become live repositories of MicroStrategy scorecards, reports, and charts, all of which are continually linked back to Intelligence Server ensuring 100% data consistency across the enterprise.

Additionally, MicroStrategy Office offers outstanding offline analysis capabilities. Microsoft Office is the most natural place for business users to format, sort, and analyze data disconnected from a network. Once back online, MicroStrategy Office refreshes the files with the latest data with just one mouse click while preserving the analytical and formatting work.

3.8 MicroStrategy Desktop

MicroStrategy Desktop™ is an advanced development and analysis interface for creating BI applications for analysts, power users, and application developers. Desktop builds the metadata objects used for designing reports, scorecards, and dashboards. It employs editors and wizards to speed-up application development without the need for additional coding. Together with MicroStrategy Architect, they are the key development tools required to create BI applications.

Desktop is an equally powerful BI interface for the most advanced analysts who aggressively investigate the data to uncover valuable insight. Desktop provides a full range of analytical functionality for reporting, data mining and predictive analysis, statistical analysis, financial analysis, mathematical analysis, set analysis, and time series analysis. Desktop offers the rich Windows interface and power of a thick-client, workstation environment.

3.9 MicroStrategy Architect

MicroStrategy Architect™ is a rapid development tool that creates the metadata objects that map the physical structure of a database to a logical, object-oriented model of the business. Architect employs wizards and graphical editors to link the enterprise's business model to the physical database tables and columns.

The abstraction of the physical database into a logical business model makes subsequent report design using Desktop extremely fast and intuitive. The logical business model abstraction also provides a layer of isolation between the physical database and the reporting applications allowing each to change independently of the other as they evolve over time. Moreover, the object-oriented nature of the metadata repository allows all changes in the logical model to propagate immediately and transparently to all dependent objects.

The abstraction of physical models to logical models allows MicroStrategy-based analytical applications to be ported across different data warehouse platforms. This has profound benefits to analytic application developers who want their applications to be easily connected to any existing data warehouse.

3.10 MicroStrategy BI Developer Kit

The MicroStrategy BI Developer Kit™ includes MicroStrategy Architect, Desktop, and six starter BI applications or analytic modules designed to dramatically speed up the development and deployment of customized BI applications. The modules are a result of MicroStrategy's 15+ years of BI experience, and represent horizontal applications common to most every industry:

- Financial Reporting Analysis
- Human Resources Analysis
- Web Traffic Analysis
- Sales Force Analysis
- Customer Analysis
- Sales and Distribution Analysis

Each module includes a data model, an extensive library of foundation metrics, attributes, and hierarchies, and many business reports. Each module can quickly be linked to existing data warehouse to rapidly produce a working BI application. The application analytics can be easily adapted and extended to the unique needs of each enterprise. This application portability is unprecedented in the BI industry, and is only available for applications built on the MicroStrategy platform.

3.11 MicroStrategy Administrator

MicroStrategy Administrator™ is a comprehensive management toolset for centrally monitoring and managing the entire business intelligence infrastructure. Administrator was designed specifically to:

- Automate the administration of thousands or millions of users
- Manage real-time operation of mission-critical infrastructure
- Analyze trends in usage and resources
- Manage application promotion

MicroStrategy Administrator includes three primary components:

1. Command Manager – enables bulk administration and maintenance of security and system configuration for large user communities using text-based commands from a graphical or command line interface.
2. Enterprise Manager – enables analysis of database usage, report usage, user statistics, and trends to facilitate performance tuning and resource planning.
3. Object Manager – provides an object management process for deploying reporting applications from development to production environments.

3.12 MicroStrategy SDK

The MicroStrategy SDK™ is a comprehensive development environment primarily used for integrating MicroStrategy functionality into other existing systems, especially enterprise portals, and customizing and extending the functionality of the MicroStrategy Web. The MicroStrategy SDK includes:

- Documentation of the API and platform functionality
- Development kits for portal, external security, and Web services integration
- Utilities and sample code the show how the APIs can be used

The SDK is a key architectural component used to fully exploit the power of the MicroStrategy platform through its documentation of the platform's open APIs. The APIs expose the complete set of functionality contained in the platform. The platform includes the following groups of API's:

- MicroStrategy Web API
- MicroStrategy Intelligence Server API
- MicroStrategy Narrowcast Server API
- MicroStrategy Office API
- MicroStrategy Transactor API

3.13 MicroStrategy Transactor

Closed-loop business intelligence applications describe the interactions necessary between BI and other systems to take immediate action on data presented in a report. Often this entails writing data back to another target IT system, and the associated steps necessary to accomplish this.

MicroStrategy Transactor™ is a workflow server with a full complement of components to design, execute, and monitor closed-loop BI applications. Transactor manages all the tasks in a pre-defined workflow to complete a transaction. For Transactor, a transaction is any sequence of steps or workflow that performs an action in conjunction with one or more IT systems.

3.14 Summary

The MicroStrategy platform architecture offers the only true, unified, organically-developed enterprise class business intelligence technology required for today and tomorrow's most demanding BI applications. Based on one, object-orientated metadata, the MicroStrategy platform consists of 11 distinct products designed to deliver the complete range of BI functionality to business users, the depth of analytical power to analysts and developers, comprehensive administration to IT administrators, and limitless extensibility to IT developers and integrators. Only the MicroStrategy platform architecture is worthy of the moniker 'Enterprise-wide BI Standard' now and well into the future.

PLATFORM COMPONENTS

4

Metadata and the Logical Model

Metadata is information that describes another set of data. In a business intelligence platform, metadata links information in a data store to business entities and rules that define a BI application. At the lowest level, metadata translates table and column information from a data warehouse into meaningful business terms. It unlocks valuable information stored within data warehouses by serving as an intuitive map to guide business users in building reports and conducting analyses. Today's BI platforms need centralized, reusable, and dynamic metadata to support enterprise–wide BI standardization. MicroStrategy provides the only metadata in the industry that is ready for BI standardization by incorporating the following architectural design tenets:

- Abstraction in a single, centralized location.
  MicroStrategy's metadata is stored in a central repository, where it stores mappings of business concepts to underlying data structures as well as layouts and report configurations. As a result, all users, regardless of the interface through which they access the application, receive a single version of the truth.

- Fully reusable well-defined metadata objects.
  MicroStrategy's metadata repository is comprised of multiple layers of well-defined objects, with each layer serving as the foundation upon which to build the next. This approach results in the broadest range of reusable objects, increasing development speed and minimizing maintenance costs.

- Dynamic, object-oriented metadata objects.
  Every MicroStrategy metadata object contains "pointers" to other metadata objects. Whenever an object is used, it is dynamically assembled from the latest versions of all the other objects contained in the object. Changes to an object automatically cascade to related objects.

- Storage in a standard relational database.
  In contrast to other vendors' proprietary metadata storage formats, MicroStrategy's metadata is stored in a relational database. This approach increases the scalability and manageability of BI applications.

4.1 A Complete Metadata is Key for Efficient Business Intelligence Processing

The MicroStrategy BI architecture incorporates six primary layers of metadata objects. Each layer represents a progressively higher level of abstraction from the physical data structures:

1. Configuration Objects
   Users, user groups, schedules, projects, and data source connections make up foundation of BI applications.

2. Data Abstraction Objects
   Logical tables, attributes, and facts translate physical data structures to the logical business model.

3. Business Abstraction Objects
   Metrics, hierarchies, custom groups, and transformations combine data abstraction objects into business definitions and organizational structures.

4. Report Components
   Templates, filters, Autostyles, and prompts are high level abstraction objects, and are part of the basic building blocks of a report definition.

5. Report Definitions
   Report definitions are highly formatted enterprise reports that include cross-tabular grids and graphs as well as scorecards and dashboards. These reports track key performance indicators and business processes.

6. Delivery Objects
   Delivery objects define personalized information distribution services that send reports results with static and dynamic information to report recipients.

The abstractions defined by these six metadata object layers enable quick and easy report creation, and distribution without the need for any programming. Objects are simply assembled directly from the metadata and resolved when the report is run. Since all MicroStrategy objects access the same metadata repository, any modifications to the metadata are automatically propagated to all objects and all reports.

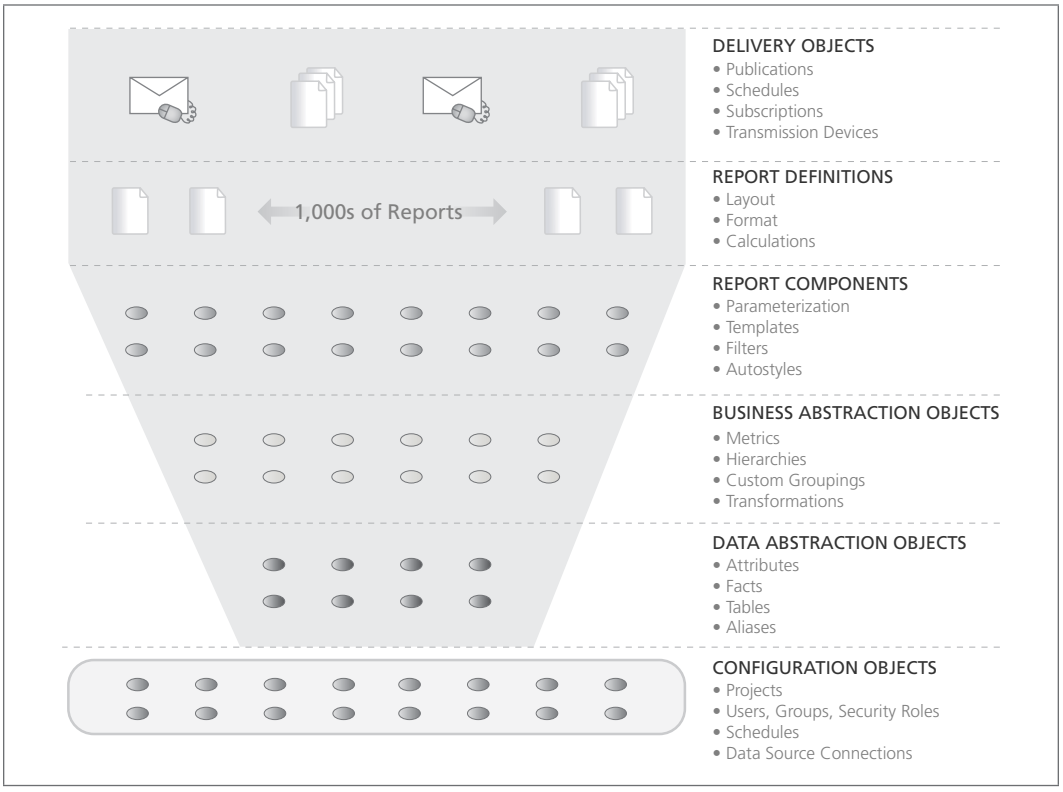


Figure 4.1 MicroStrategy's metadata speeds BI application development, and eases maintenance burdens with its object-oriented, reusable, and fully dynamic metadata.

4.2 Configuration Objects

Configuration objects serve as the framework for building BI applications, and are shared across projects. They define the scope of BI applications built on the MicroStrategy BI platform by encapsulating the accessible data sources, and maintaining users and their groups. Business definitions relevant to the entire organization, such as Security Roles and Schedules, are persisted at this level.

Building BI Applications Using Projects

A project is a group of related business definitions serving a common purpose within an organization. Together these business definitions serve as the building blocks for a BI application. By grouping them under a project, properties can be defined on top of all these related business definitions for administration and control. Each project can have its independent:

- User to Security Filters mapping
- User to User Authorization mapping
- Combination of Data Sources
- Usage Governors

Creating Users and Their Groupings

Each user within an organization can access a different set of BI applications. By placing user objects at the Configuration Object level, they are shared across projects. Within the MicroStrategy BI platform, users are grouped into different users groups for efficient user management. Each user group can contain multiple users, and each user can belong to multiple groups. User objects contain the following user information:

- Login
- Full Name
- Description
- Password
- Password Properties
    - User cannot change password
    - User must change password at first login
    - Expiration Date
- Project Access
    - Security Role used
    - Privileges Granted
- Security Filter per Data Source
- User Groups membership
- Authentication Parameters for NTLM, Database, and LDAP Authentication

Representing Data Sources and Their Access Pathways

While each BI application within the MicroStrategy BI platform can access multiple data sources, information on their characteristics and connectivity is stored once and reused. Database objects encapsulate details on the nature of the relational, cube or other data source along with the data source-specific MicroStrategy platform optimizations. Database Connection objects contain details on how connectivity to various data sources is achieved and maintained while Database Logins hold information on how the Intelligence Server should authenticate to these data sources. Together these objects define the scope of data accessible by all business definitions within a same particular MicroStrategy metadata repository.

Within each project, connection mappings allow different users to access each data source using different Database Login objects. When users share the same Database Connection and Database Login objects, the same connection to the data source is reused for these users' queries.

Security Roles and Schedules

Each user within an organization has a different role, and should thus have visibility to different aspects of its operations. For example, a Human Resources manager has access to all employee hiring records but may be restricted from projected revenue data. Mirroring business operations in an organization, this same HR manager can have different privileges within the BI application for Human Resources and the BI application for Sales Forecasting. For ease of maintenance, these privileges are bundled under Security Role objects within the MicroStrategy BI platform.

Time and event-based schedules allow reports executions and administrative tasks to occur at any time of the day or week. Actions across different projects can share schedules for simultaneous processing.

4.3 Data Abstraction Objects

Data abstraction objects map the physical data model of the data source into a logical business model using business terms and rules. Data abstraction objects serve as the building blocks for business abstraction objects and, ultimately, for all reporting, analysis, and monitoring.

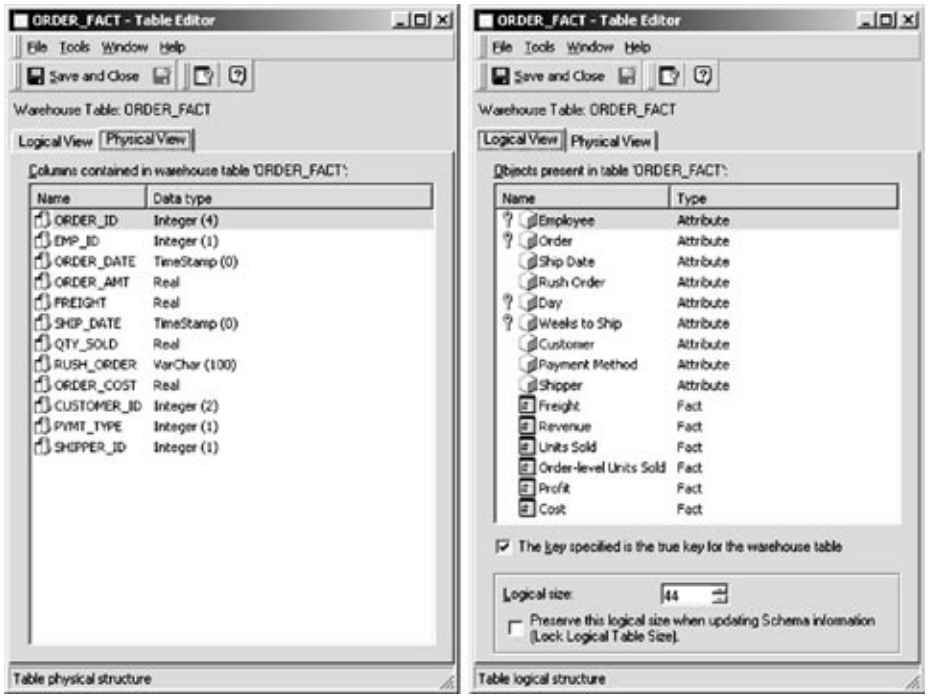


Figure 4.2 The metadata stores a physical and a logical view of a database table.

Physical and Logical Models of Relational Data Warehouses

The MicroStrategy architecture supports a wide range of physical data warehouse schemas, including star and snowflake schemas. These warehouse schemas can contain sparse aggregate tables, partition tables, and split fact tables. MicroStrategy also provides direct access to operational data stored in third-normal form. When working with a traditional data warehouse, MicroStrategy Architect retrieves table, view, column, and datatype information directly from the database system catalogs. The logical business model, including the relationships between the

business entities, is created from this information. The physical data structure and logical business model are presented in two graphical views within MicroStrategy Architect:

1. The physical table structure displays the data warehouse tables, columns, and data types.
2. The logical table structure displays the table and column information in the form of attributes and facts. The relationships between the attributes are displayed in a multi-dimensional view.

Logical tables can also be defined from hand-coded freeform SQL statements that combine data from different physical tables and views in the data warehouse. This approach provides many of the benefits of database views while working exclusively within MicroStrategy's object-oriented metadata.

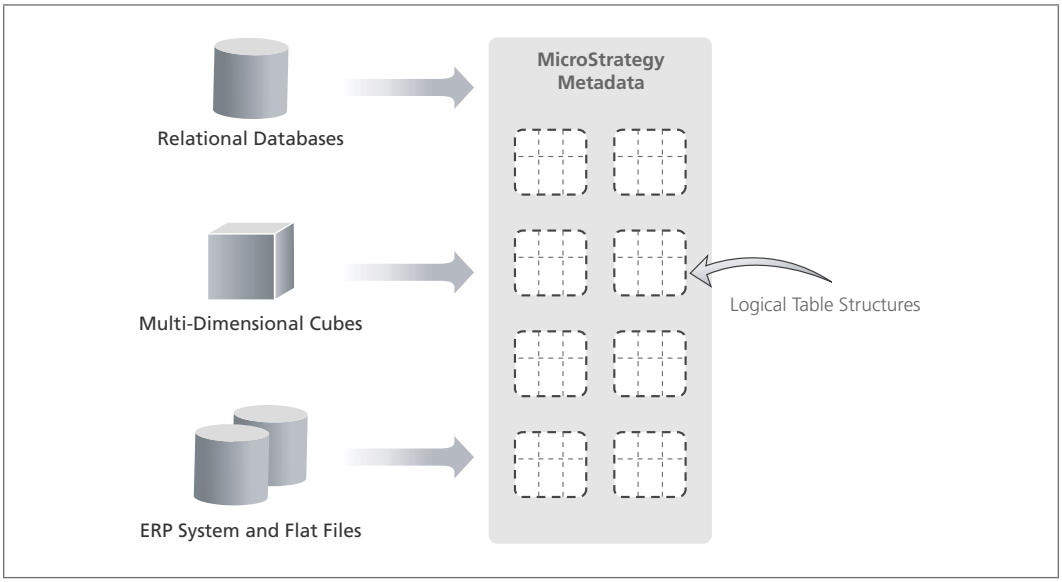


Figure 4.3 Information from all data sources is uniformly mapped in the MicroStrategy metadata.

Facts Are the Basic Building Blocks of Business Metrics

A fact is a table column, or an expression combining multiple columns, that contains raw or aggregated values such as amounts, counts, quantities, forecasts, and other measurable business calculations. A fact stored as a data abstraction object in the MicroStrategy metadata is represented as:

- A single column, e.g. DOLLAR_SALES
- A single column with an aggregation function, e.g. SUM (DOLLAR_SALES)
- Several columns combined using functions and expressions, e.g. SUM(PROFIT / DOLLAR_SALES)

Attributes Give Context to Facts

Attributes are defined by a column or columns in one or more tables, and provide meaningful context to facts. Attributes are commonly displayed on reports. They also serve to set the aggregation level for calculations in a report, and for filtering the data that is retrieved from the data warehouse. Examples of attributes are regions, stores, customers, and months.

Attributes can have many descriptors or forms as they are known in MicroStrategy. An attribute form is one of several columns that are different representations of the same attribute. For example, the attribute Product might have the following forms – an ID, short name, long description, SKU, and image.

Attribute elements are the actual data values of an attribute stored in the data source. They are retrieved and cached by MicroStrategy when running reports and analyses. For example, New York and Paris are elements of a City attribute.

Abstractions for Relational, Multi-Dimensional, and Operational Data

For other data sources, MicroStrategy's BI architecture translates the physical data structure – whether it is a database table, a multi-dimensional cube, a collection of ERP tables joined together, or an Excel spreadsheet – into a logical table structure. These logical table structures form the basis for defining attributes, facts, and higher level metadata objects.

MicroStrategy accesses cube-based data sources, and automatically converts the multi-dimensional models into logical MicroStrategy metadata objects. For example, SAP® InfoCube, QueryCube, and ODS objects can be used as MicroStrategy metadata objects. SAP levels, attributes, and structures are imported as MicroStrategy attributes, and key figures are imported as metrics. MicroStrategy also joins data across SAP BW InfoCubes, QueryCubes, and other MicroStrategy data sources, and can access multiple instances of SAP BW at once.

Attributes and metrics from all data sources can be mapped to each other. As a result, data from all these sources can be joined, and transparently presented in a single document to business users. Calculations involving metrics from different sources are also possible.

Freeform SQL maps operational data to MicroStrategy metadata objects. For any data source, custom SQL can be provided to extract subsets of information, and mapped to existing or new attributes within the metadata. Any data source accessible via ODBC can be integrated into the MicroStrategy metadata.

Flexible Schema Combinations Make Complex Business Modeling Simple

Complex business models that are not stored as standard data structures are simple to define in MicroStrategy. In addition to basic column mapping, as described above, facts, attributes, and attribute forms can use any function supported by the database platform to combine and extend data in the data warehouse. This capability eliminates the need to change the physical data structure in order to support an organization's reporting and analysis requirements.

For example, only the customer's date of birth is stored in the database. There is no need to create a new column in the data warehouse for customer age, and periodically update this column. Instead, the attribute definition for the customer age is an expression that dynamically calculates the difference between the system date and the customer's date of birth.

4.4 Business Abstraction Objects

Business abstraction objects define the business terms and rules. They serve as the building blocks for other business abstraction objects and for all reports, analyses, and documents.

Metrics are the Business Measures

Metrics represent business measures and key performance indicators. Metrics consist of facts, attributes, or other metrics combined with a formulaic calculation. This formula may include aggregation functions such as "Avg", "Sum" or "Count," arithmetic operators (+, -, *, /) or other advanced functions (e.g., data mining, financial, mathematical, OLAP, statistical). A metric can contain 3 additional properties:

- Dimensionality
  Metric Dimensionality determines the level of aggregation of a metric. The default aggregation level is based on the attributes contained in a given report. The aggregation level of a metric can be changed to force aggregation at a specified level. This flexibility is especially useful if business metrics need to be calculated at the same level regardless of the level of reports that use them.

- Filter
  Metric Filters determine any restrictions on the data used in the calculation. A metric filter may be defined as an embedded component of the metric or as a link to existing filters, maximizing object reuse, and reducing administration.

- Transformation
  Transformations may be associated with a metric to enable time series analysis as described below.

Predictive Analysis Scoring Metrics

MicroStrategy's BI platform incorporates sophisticated data mining algorithms that define powerful predictive analytic metrics. Data mining is the process of using statistical analysis, modeling techniques, machine learning, and other technologies to identify hidden patterns and predictive information in a data warehouse.

Predictive models from third-party data mining tools are imported using a Predictive Model Markup Language (PMML) representation of the model. A scoring predictive metric and confidence predictive metric is generated. These metrics are stored in the MicroStrategy metadata, and used like any other metric.

Transformations Add Great Flexibility to Facts

Transformations are reusable business abstraction objects that are embedded primarily in metrics for time-series analysis calculations. They automatically display information for changing dates and time periods without constantly needing to redefine metadata objects. For example, many reports display current period information along with prior-period and period-to-date information. In this example, transformations handle the variable time periods automatically regardless of when the reports are executed within the same object definitions. Two types of transformations are supported in MicroStrategy:

- One-to-one transformations compare one period to another, for example, "Last Month" and "This Month Last Year".
- Many-to-many transformations define period-to-date and last-n-period metrics, for example "Month-to-Date" or "Last Four Weeks".

Hierarchies Align the Attributes into Business Dimensions

A hierarchy is a logical group of attributes that defines a relationship between them. A hierarchy describes an ordered path through the attributes from a summarized level to more detailed levels. Hierarchies are used to browse data, and provide a navigation path when drilling from one attribute to another on a report.

In MicroStrategy, hierarchies also serve as placeholders for an attribute in a report, with the specific attribute being determined at run-time. There are two types of hierarchies:

- System Hierarchy
  The system hierarchy is a direct map of the logical table structure stored in the metadata, and shows the true relationships between all attributes in the reporting application. There is only one system hierarchy for each MicroStrategy application.

- User Hierarchies
  MicroStrategy allows a greater range of hierarchical paths than those found in the physical database. User hierarchies define relationships between attributes based on the business rules of the reporting application, and they guide user analysis or provide shortcuts to common analytic workflows. Attributes can be reused in different user hierarchies.

Custom Groups Merge Data Elements into Virtual Attributes

Custom groups define arbitrary groupings of attribute elements in a report, and represent data groupings that do not exist in the data warehouse itself. They are defined either by using filters or by specifying metric bands to group certain attribute elements together. User-defined descriptions identify each row of custom group elements.

Custom groups are commonly used for financial statements. For example, each row in a profit and loss report aggregates groups of ledger accounts and sub-accounts into a single value. Custom groups define these account groups for the various sections of the report: revenues, costs, and expenses.

Consolidations Extend the Definition of Custom Groups

Consolidations serve a function similar to that of custom groups in that they act as user-defined virtual attributes on reports, but their definition is restricted to filters on actual attribute elements that make up the group. With consolidations, additional groups can be created by performing arithmetic operations between any of the other groups in the consolidation.

4.5 Report Components

Report components are dynamic, reusable metadata objects built with the data abstraction objects, business abstraction objects, and other report components. These three layers of metadata objects are combined to create new reports and analyses in both MicroStrategy Desktop and MicroStrategy Web. In all cases, the metadata objects retain their reusable and dynamic characteristics, and a change to a single object in the metadata is automatically reflected in all other dependent metadata objects.

Filters Focus the Data of a Report

A filter specifies the conditions that data must meet to be included in a report or metric. Filters limit the data that is extracted from the data source, and focus the data presented to the end user. A filter is essentially a logical expression that combines any metadata object, mathematical and comparison operators, and constant values and lists.

Filter expressions can be combined using Boolean operators and nested to create sophisticated conditional clauses. For example, to retrieve data for New York and San Francisco for the fourth quarter of 2005, the filter would be:

(City='New York' OR City='San Francisco') and Quarter='2005Q4'.

Prompts Give Users Control Over Report Content

Run-time prompts provide report input parameters that control most aspects of a report, and give the user significant ad hoc reporting capabilities. Prompts enable the personalization of reports from a single report definition, reducing the number of objects stored in the metadata repository. There are four types of prompts in MicroStrategy:

- Filter prompts provide filtering criteria at runtime, and qualify the data set from the data warehouse that will be included in the report results.
- Object prompts let users choose which attributes and metrics to include on a report at run-time. They can also be used to build metadata objects, particularly metrics, providing enormous flexibility from a single report definition.
- Value prompts allow the input of alphanumeric and date values that specify constants for filtering criteria and metric calculations.
- Level prompts dynamically control the aggregation level (dimensionality) of metrics.

SAP Variables are Imported as Prompts

SAP BW uses variables to specify filtering parameters at run-time. As part of MicroStrategy's certification with SAP BW, the MicroStrategy BI platform automatically translates SAP BW variables into reusable MicroStrategy prompts. Each MicroStrategy prompt contains a dynamic pointer to the SAP BW variable; changes to the definition of an SAP BW variable automatically ripple through to MicroStrategy metadata objects.

Templates Support Any Layout for a Report

A template specifies the layout and formatting of the data on a report. Attributes, metrics, custom groups, consolidations, and prompts can be placed in rows, columns, or pages on the template grid. There is no theoretical limit to the number of objects that can be placed on a template. Besides being able to define the presentation characteristics, such as font, color, alignment, and number formats, additional report characteristics such as subtotaling, metric thresholds, and graph properties are also stored in templates.

Autostyles

An Autostyle defines a reusable report format. The MicroStrategy BI platform contains more than 35 Autostyles and new Autostyles can easily be added to match corporate standards, business reporting requirements or personal formatting preferences. New Autostyles are immediately available to all MicroStrategy users. Default Autostyles can be specified for projects, reports, and individual users.

4.6 Report and Document Definitions

A report is simply a user request for specific, formatted data from all enterprise data sources. Reports gather, consolidate, search, analyze, and convey information. MicroStrategy reports take on any form and role, including predictive analysis, cross tabular grids and graphs, operational reports, customer invoices, executive dashboards, scorecards, and proactive alerts. The report definition includes all information necessary to calculate, place, and format the information.

Documents are a compilation of reports. Report definitions are carried over when report data sets are placed in documents. Multiple reports datasets combined with multiple formats in a single source serve to provide multifaceted and comprehensive insight into the business. A high degree of formatting and layout control further increases the flexibility of the MicroStrategy BI platform in meeting any reporting needs.

MicroStrategy Reports are Highly Interactive

Interactive MicroStrategy reports consist of two components: a template and a filter. MicroStrategy reports are fully interactive, allowing users to toggle between grid and graph mode, rearrange report layouts by pivoting rows, columns and pages, change the formatting, sort the columns, add subtotals, and create new calculations in the report.

Each report can access any of the other metadata objects at all levels. The reusable, dynamic metadata lets business users perform investigation and analysis on data that was not included on the original report. For instance, a weekly product sales report can be transformed into a list of customers most likely to respond to a direct marketing offer for the top 5 selling products by drilling to any attribute in the data warehouse and applying filtering conditions. These navigation paths do not need to be defined in the metadata at all. All calculations and relationships are handled automatically by MicroStrategy based on the object-oriented metadata objects.

At any time, a new report can be saved as a new metadata object. This new object stores dynamic links to all the data and business abstraction objects, report components, and formatting and layout settings.

Report Services Documents Provide Pixel-Perfect Layout

MicroStrategy Report Services lets users create desktop-publishing-quality reports on top of their existing data using an intuitive design interface. Users create reports with pixel-perfect precision by simply dragging-and-dropping, and then formatting reports for boardroom quality presentation. MicroStrategy Report Services also delivers reports in Portable Document Format (PDF), ensuring printed reports have the same quality as on-screen reports. After placement and adjustments of document elements, formatting is defined in segments of bands and zones within bands, giving maximum flexibility in creating sophisticated presentations of data.

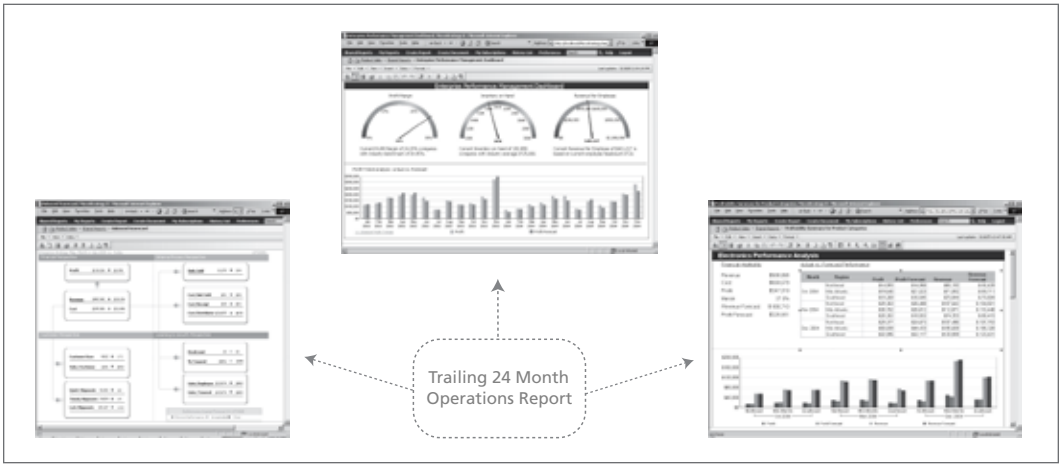


Figure 4.4 With MicroStrategy's reusable metadata, a single report definition can be used in any number of Report Services documents.

4.7 Delivery Objects

Delivery objects build upon report definitions to deliver personalized, subscription-based information to business users. MicroStrategy Narrowcast Server combines report definitions, unstructured text and data, and custom formatting with schedules and subscription lists to offer a massively scalable report delivery solution.

Report Delivery Services are Automated and Highly Customizable

A report delivery service contains all the information required to deliver personalized and timely information to all business users. A Service consists of four delivery objects:

- Publications – store the message content
- Schedules – frequency or exception condition for generating the content
- Subscriptions – users who receive the content
- Transmission Device – delivery method for the content

Publications Define the Content and Formatting

A publication specifies all content for the delivery service, and can include both static content and dynamically generated information. A publication is composed of any combination of MicroStrategy reports and documents, images, static text and non-relational data. Publications are the HTML, Excel, or PDF files that are delivered.

Time or Event Based Schedules Trigger Service Deliveries

A schedule sets the times or frequencies that a delivery service is executed, and represents a recurring pattern – not a fixed date – on which a service is executed. Schedules may be time or event based. Different schedules may be associated with a single service, and may be reused across any number of services.

Self-Service or Administrator-Defined Subscriptions

A subscription specifies who receives a specific delivery. Subscriptions may be personalized so that a user's prompt answers will automatically be applied when the delivery services executes. Subscriptions may be self-service or administrator-defined.

- Self-Service Subscriptions
  Business users subscribe to receive services through MicroStrategy Web and through the Subscription Portal. Reports can be delivered on a scheduled basis, with or without personalization, to the location of the user's choice in a variety of formats (HTML, PDF, Excel, Print copy).

- Administrator-Defined Subscriptions
  Delivery service subscriptions can be set up centrally in order to accelerate deployment to thousands of users, ensure each user receives the correct reports, or reduce unnecessary network traffic.

Transmission Devices Specify the Delivery Channel

Transmission devices specify the communication mediums and format used to deliver the requested publication, at the appropriate time and with any relevant personalization, to all the correct users. Narrowcast Server contains transmission modules that deliver services directly to portals, email, print servers, and file servers. Transmission modules to other devices can be added to deliver information to virtually any corporate system.

4.8 Comprehensive Documentation of Metadata

MicroStrategy provides comprehensive, interactive documentation of the metadata. From any object, all descriptions, definitions, formulae, object dependencies, and other properties are viewable. Extracts of the documentation can be published to ensure all users possess an intuitive understanding of the data model, reports, and other metadata objects.

Graphical Hierarchies Illustrate Relationships Between Logical Business Terms

MicroStrategy Architect's graphical hierarchy viewer displays the relationships between business attributes and underlying data elements. These graphical hierarchy views may be printed or saved, and published to business users so that they can easily understand the business model and attribute relationships.

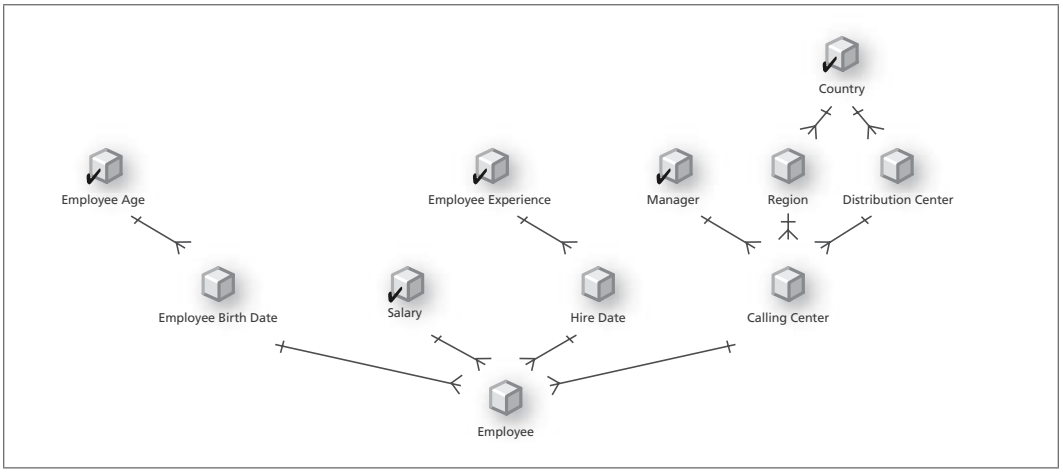


Figure 4.5 Viewing hierarchies graphical shows the relationships between the attributes in the hierarchy.

Interactive Project Documentation Gives a Detailed View of BI Applications

MicroStrategy Architect contains a Project Documentation Wizard that generates detailed documentation of the metadata. The wizard allows fine-grained control of the objects and the amount of detail to include in the documentation. The Wizard's outputs HTML, enabling navigation through the complete metadata structure using the object dependencies. The output files may be printed, published on a Web server or shared drive, or stored in a configuration management system to track changes that occur in the metadata.

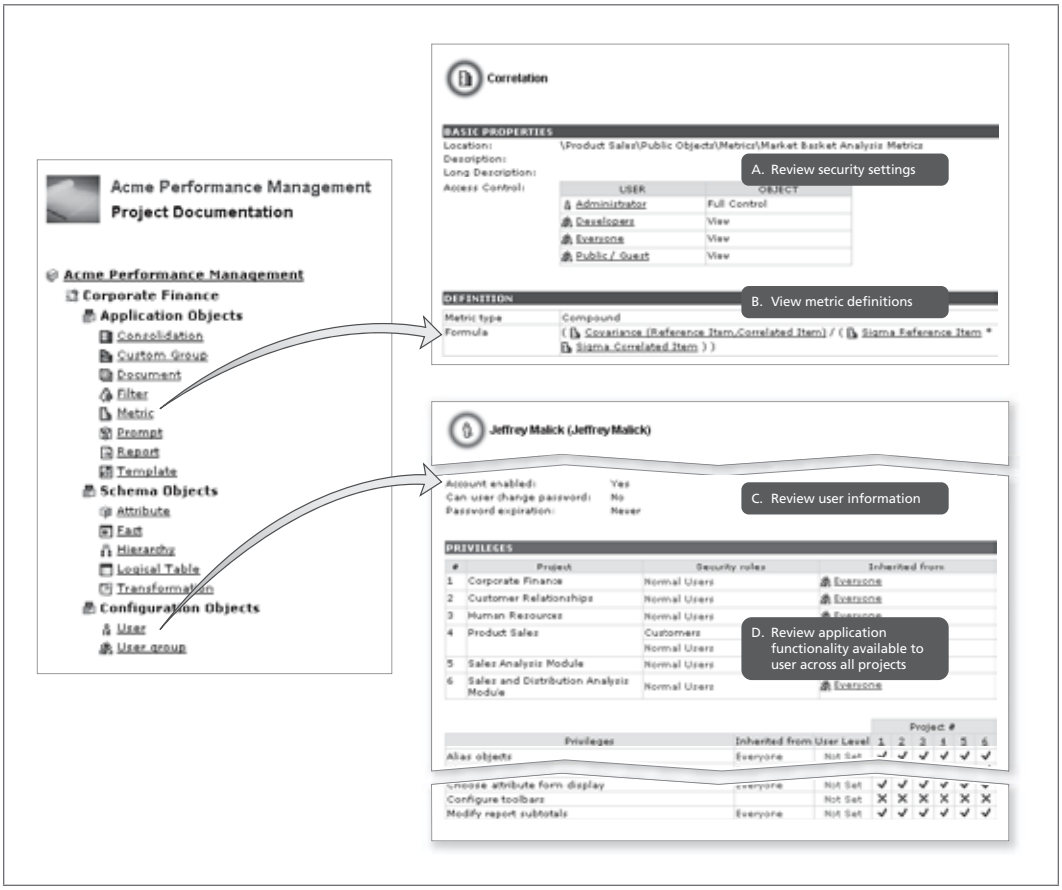


Figure 4.6 Metadata documentation includes important details about the metadata objects.

4.9 Benefits of Metadata Design

MicroStrategy designed and built its unparalleled metadata architecture based upon the following goals:

- Provide an abstraction of the physical data structures, and store these logical objects in a standard relational database
- Provide a single, centralized, and shared metadata for all reporting, analysis and monitoring
- Provide a fully reusable metadata
- Provide a truly dynamic and fully object-oriented metadata

These design principles result in tangible benefits for organizations that use MicroStrategy's BI platform, including:

- Intuitive reporting, analysis, and monitoring for all business users, and across all data sources
- The most efficient report development process
- Maximum maintainability and consistency across the entire enterprise

Abstraction from Warehouse Provides Intuitive Reporting, Analysis, and Monitoring for All Business Users Across All Data Sources

MicroStrategy's metadata abstraction translates the physical structure of a data source into easy-to-understand business terms, simplifying the deployment of reporting, analysis, and monitoring applications. There is no need to understand query languages such as SQL, be familiar with the underlying data model, or even be knowledgeable about the actual data source.

The different abstraction layers ensure that changes to objects in one layer do not materially affect other layers, and that changes are automatically propagated to all dependent metadata objects.

A comprehensive metadata is perhaps the single most important requirement for a true BI platform. It provides a way for a broad community of users to access valuable enterprise data using business terms, rules, and logic without requiring any technical skills or significant training.

Highest Number of Reusable Metadata Components

Most competing BI tools focus on developing individual reports as quickly as possible, neglecting the reusability of the report components, and sacrificing the ability to create a large number of reports efficiently. For true enterprise-class scalability, the BI architecture should allow each report to be developed more quickly than the last report, and should minimize the number of reports that need to be explicitly defined in the first place.

MicroStrategy has the broadest range of reusable objects in its metadata repository, from the data abstraction layer all the way to objects on reports. Even reports themselves can be re-used as filters for other reports, and as datasets for documents. Other BI technologies rely far less on reuse, and far more on each developer's ability to create the same components over and over again for use in each individual report.

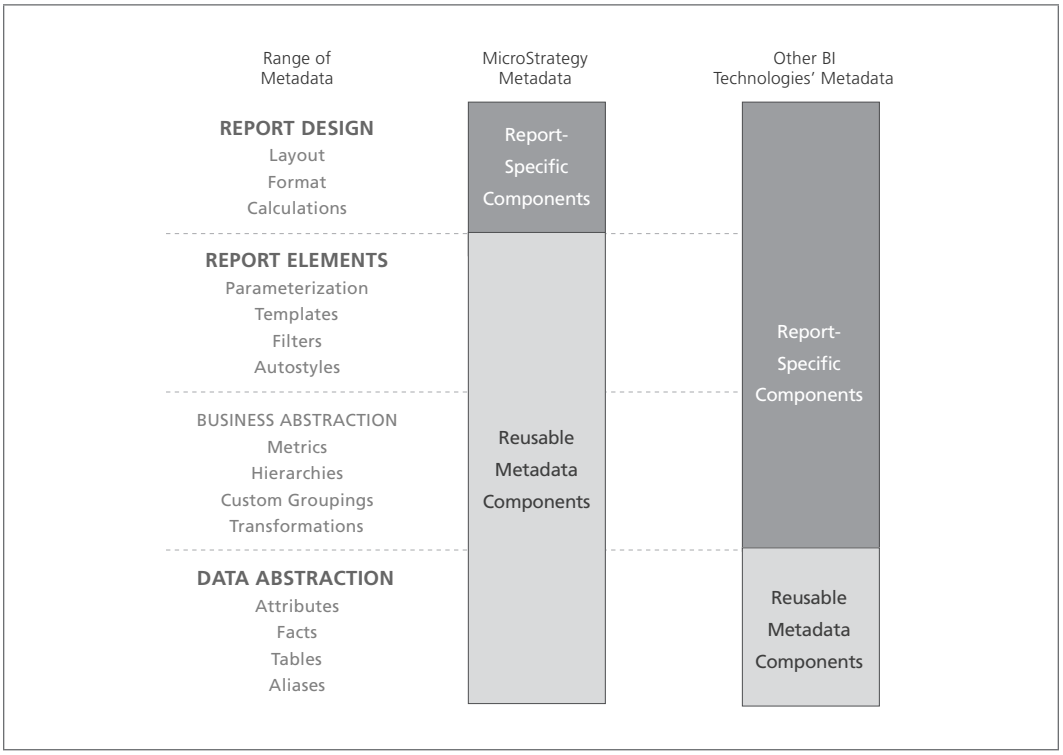


Figure 4.7 The MicroStrategy metadata architecture includes far more reusable components than any competing architecture, delivering much greater consistency across reports developed by different people.

Reusable Metadata Across All Styles of Business Intelligence

MicroStrategy's metadata object reusability spans all BI styles and applications. For example, a metric that uses a data mining algorithm is immediately available for alert notifications, operational reports, corporate dashboards, and ad hoc queries. With other BI tools, most of the business abstraction and report components need to be recreated separately each time a new application or style of BI is implemented. In addition to avoiding this extra development effort, MicroStrategy's reuse of objects ensures consistency across all BI applications and users.

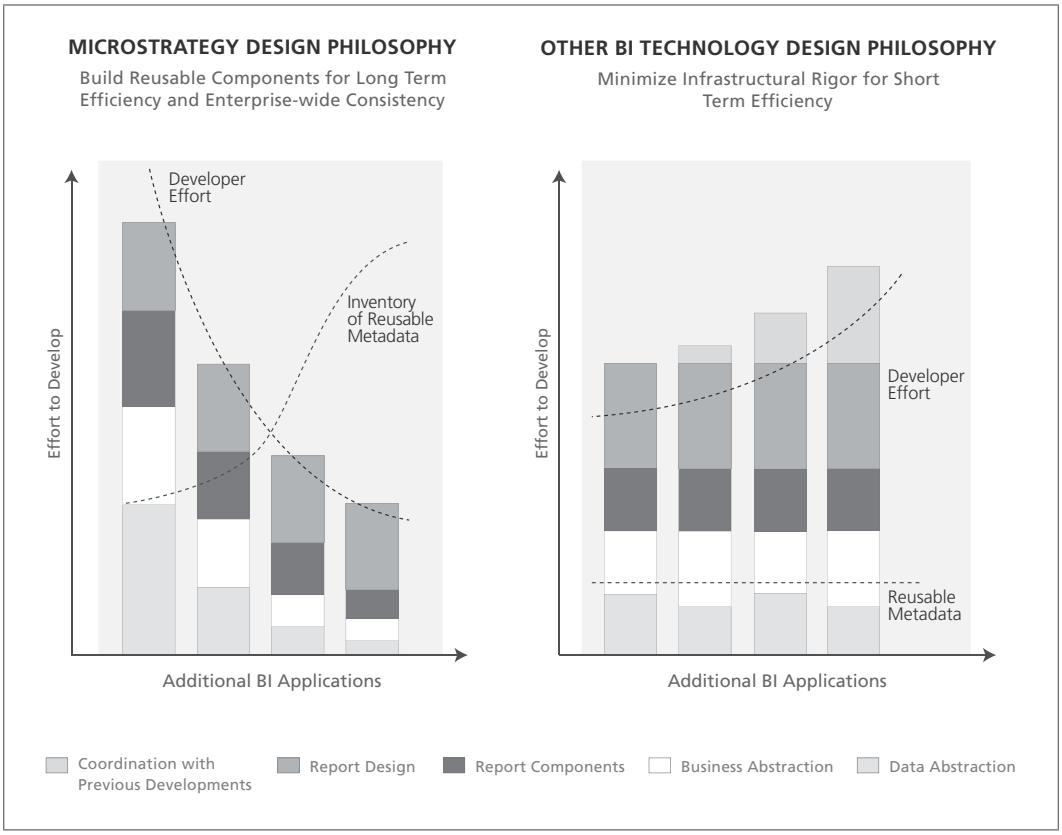


Figure 4.8 With MicroStrategy, new applications or styles of BI are deployed more quickly than the last because each application builds on all previous development. Other BI technologies actually require increasing effort to ensure consistency with all previous reports.

Dynamic Object Assembly Maximizes Maintainability

All MicroStrategy metadata objects are constructed using references rather than copies to other metadata objects. When an object is used, all referenced objects are automatically accessed and merged into that object instance. For example, when a report is run, MicroStrategy dynamically assembles the latest versions of all underlying reusable objects into the report definition, and then generates optimized SQL or MDX syntax based on the assembled objects.

Any change to an object is automatically and instantaneously reflected in every metadata object that references that object. For example, if a metric is used to define a filter, and the filter is used in a prompt, then any change to the original metric will be automatically reflected in both the filter and prompt.

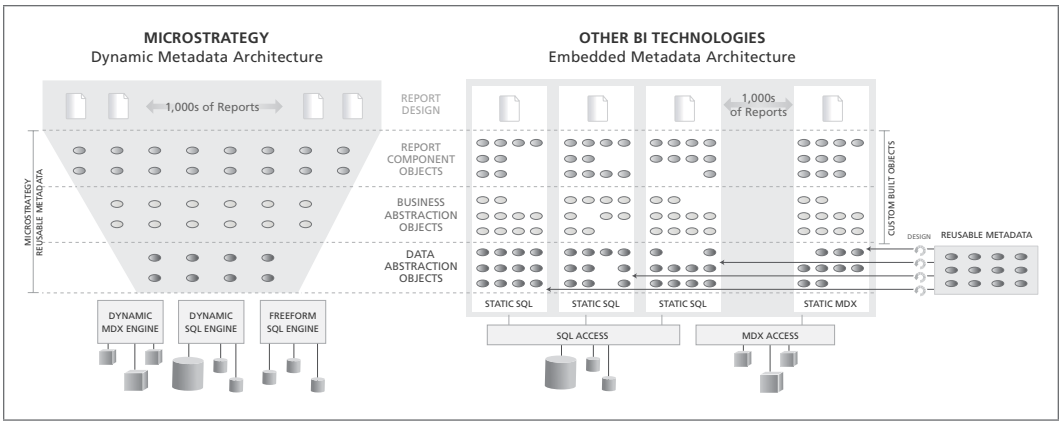


Figure 4.9 With MicroStrategy's Dynamic Metadata Architecture, each report definition contains "pointers" to metadata objects that are dynamically shared with other report definitions. By contrast, other BI technologies depend on embedded metadata where each report definition contains redundant copies of shared metadata.

Reusable Objects Ensure Consistency and a Single Version of the Truth

In the MicroStrategy metadata, there is only one definition of any object regardless of how many times it is used in different reports and objects. As a result, there are fewer objects to maintain. Additionally, since metadata objects map corporate data to the corporate business model, they ensure a single definition of any business entity in all BI applications deployed throughout the enterprise.

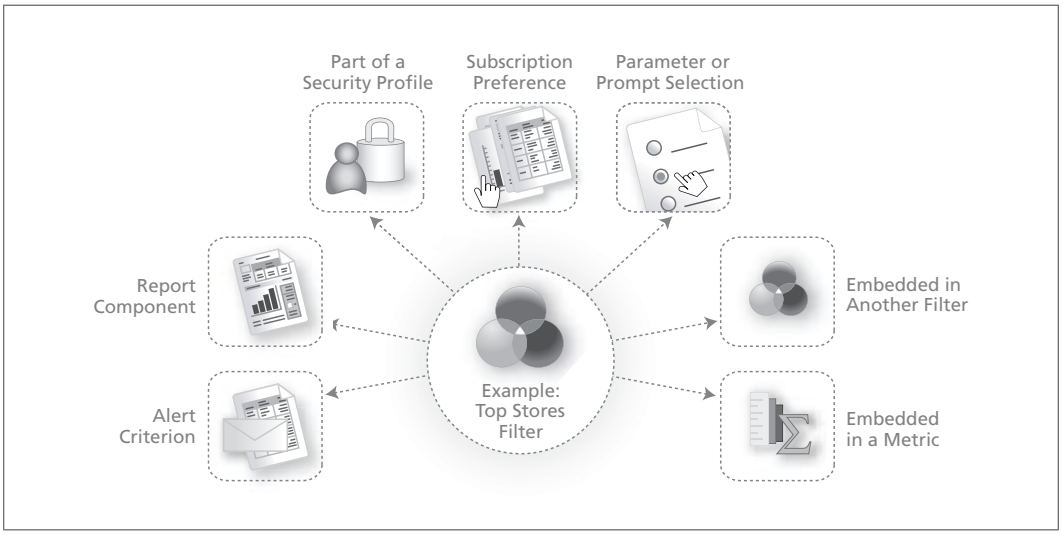


Figure 4.10 MicroStrategy's recursive object definitions ensure that object updates are necessary in only one place. All the dependent objects are updated immediately, transparently, and without any additional effort by the developer.

For example, a Top Stores filter, defined as "All stores whose sales exceed the average by 2 standard deviations," can be reused in many other objects including: a top stores grid report, a top stores graph report, a metric, a filter, or a prompt.

Ad Hoc Querying with Highly Effective Drilling and Report Parameterization

MicroStrategy's dynamic assembly of report components delivers sophisticated, automated drilling anywhere in the data warehouse. A drill action incorporates the destination attribute and additional filtering criteria into the report definition dynamically before generating optimized SQL or MDX. In-depth investigative analysis is available from any starting report in a BI application.

Virtually any object can be parameterized and selected by the user at run-time using prompts. A single report definition caters for a large user population by allowing them to select the filter criteria, the attributes and metrics in the report, and even define dynamic calculations. This MicroStrategy capability reduces metadata sprawl, and speeds development as users are able to help themselves rather than rely on report designers to create custom reports.

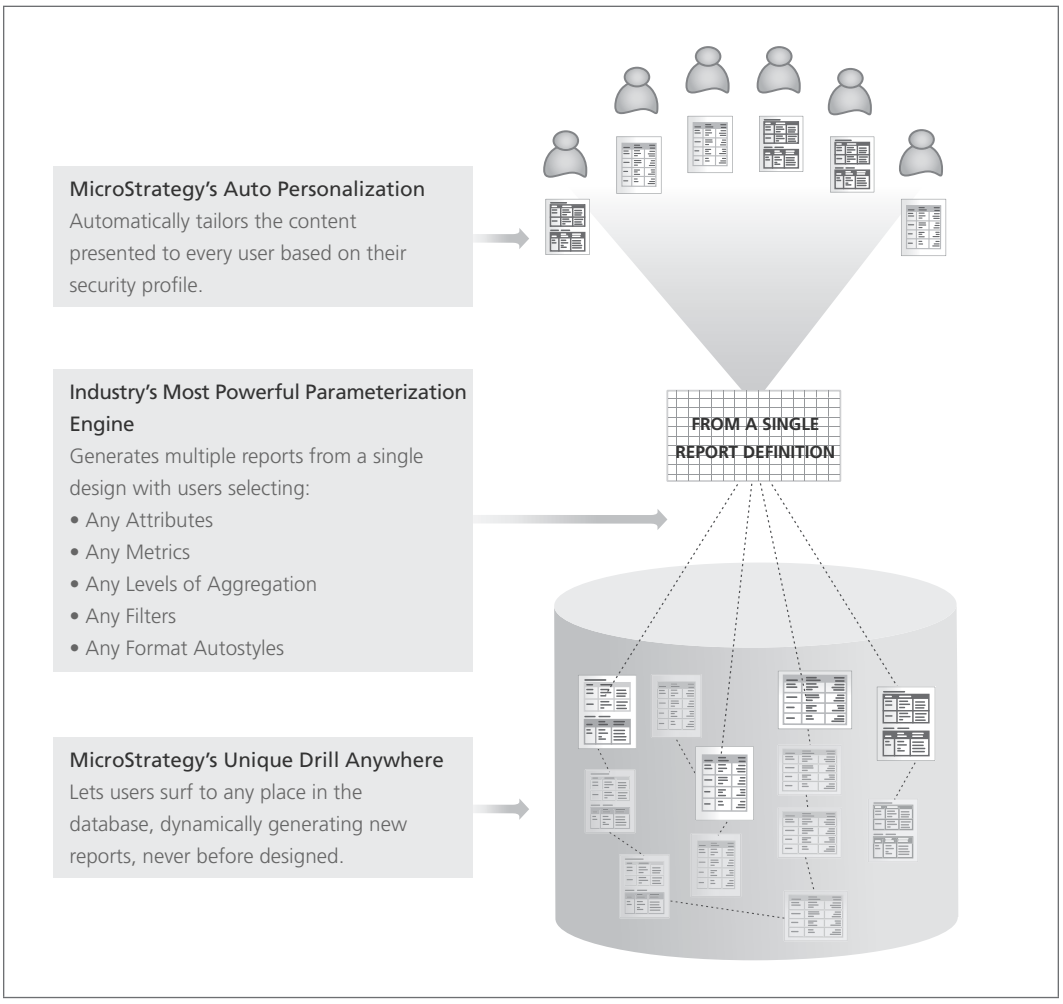


Figure 4.11 The MicroStrategy architecture routinely reduces the number of stored report definitions through automatic personalization, parameterization, and drill-anywhere.

4.10 Summary

MicroStrategy's metadata architecture, built and refined over many years, and based on robust design principles, provides access to all enterprise data using familiar business terms, rules, and logic. The reusable, object-oriented and dynamic nature of the metadata provides the most efficient BI report, and application development processes while maximizing maintainability and enforcing enterprise-wide consistency.

5

MicroStrategy Intelligence Server Architecture

The Business Intelligence (BI) standard for enterprises must provide a level of architectural and managerial sophistication that is equal to enterprise operational systems. The MicroStrategy BI platform service-oriented architecture is designed to power enterprise deployments while retaining ease and flexibility in satisfying all business needs. It offers the scalability, sophisticated analysis, extensibility, ease-of-use, performance, reliability, maintainability, and the 24x7 availability required by mission critical monitoring, reporting, and analysis applications.

The architecture is comprised of three main pillars; MicroStrategy Web, Narrowcast Server, and Intelligence Server, with Intelligence Server as the backbone powering the entire platform. MicroStrategy Web and Narrowcast Server are the major nodes from which user's access information, and from which information is distributed to users. The design tenets for the MicroStrategy architecture meet extremely stringent requirements for architectural integrity and manageability, including:

- Unlimited data scalability
- Unlimited user scalability
- Full range of analytical processing
- Any RDBMS or other data source
- Any user interface
- Enterprise deployment ready
- Highly manageable
- Single version of truth
- Automatic fault tolerance
- Open Standards conformance

To meet these challenges, Intelligence Server contains a group of well-defined, modular, and reusable components coordinated by a centralized management engine. This federated architecture within Intelligence Server provides maximum scalability, performance, and analytical power. Major highlights include:

- 64-bit technology
- Optimized for any database
- Sophisticated analytic power
- Multi-level secure shared caching
- Extensive governing parameters
- Adaptive, robust security
- Automatic resource allocation
- Flexible deployment options

The result is an offering unrivaled in the BI industry in offering enterprise–class BI capability along with the operational integrity of mission-critical ERP systems.

5.1 Powering the MicroStrategy BI Platform

In the MicroStrategy Business Intelligence (BI) architecture, report processing is performed by Intelligence Server, which manages live and scheduled report requests, services these requests either from cache or from the data sources, and adds analytical insight and formatting to the results. Intelligence Server is the core processing and management tier for the MicroStrategy business intelligence platform, and is linked to the other platform components using a highly optimized communication framework.

The communication framework handles the information flow between different platform components, and provides secure user access, efficient report delivery, and complete data interactivity. Intelligence Server manages all report requests uniformly from any requesting source, including:

- MicroStrategy Web and Web Universal
- MicroStrategy Desktop
- MicroStrategy Office
- MicroStrategy Architect
- MicroStrategy Narrowcast Server
- MicroStrategy Administrator
- Third-party applications accessing the MicroStrategy APIs

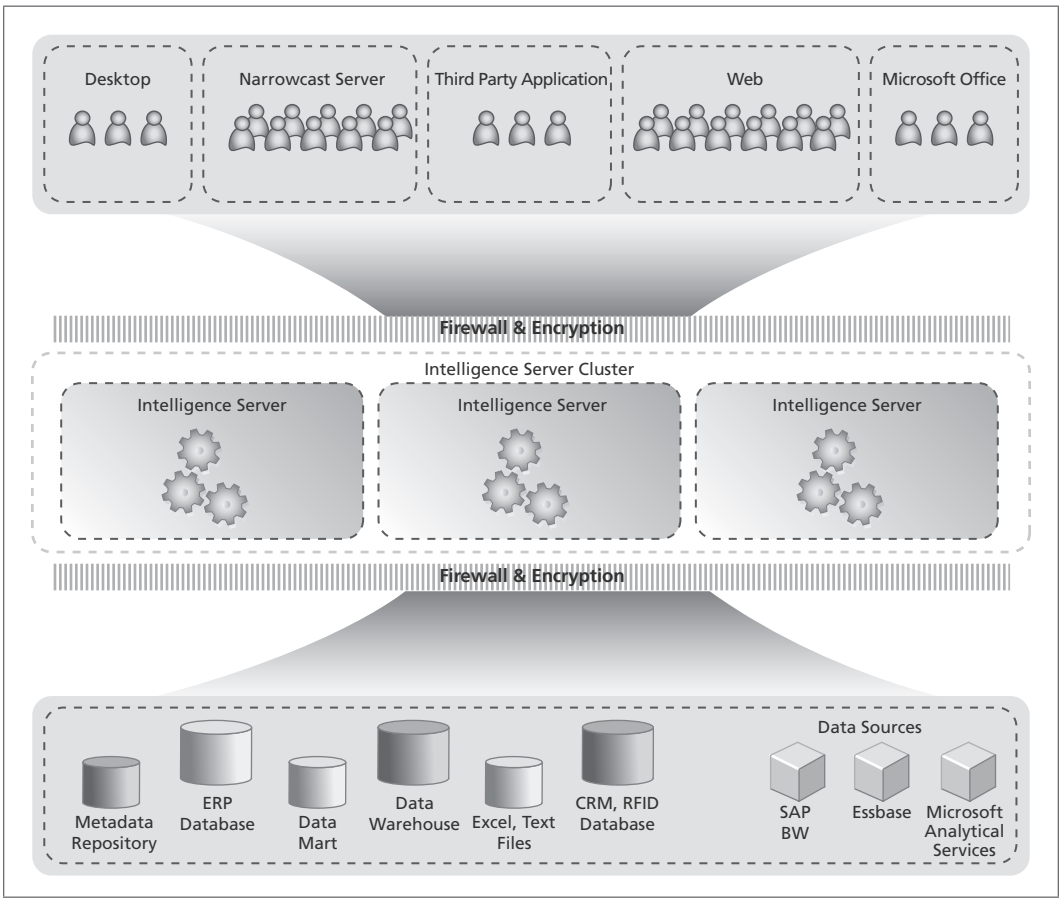


Figure 5.1 MicroStrategy Intelligence Server is the core of the MicroStrategy BI architecture.

Centralized BI Server Maximizes Productivity

This centralized architecture provides ease of maintenance, ease of management, integrated security, and one version of the truth to all users regardless of the interface. The entire BI system can conveniently be centrally controlled from Intelligence Server. In addition, Intelligence Server is reliable and fault tolerant, with automatic failover and fallback in the event of an unexpected fault. This architecture minimizes administration and maintenance, and maximizes security and performance of the business intelligence system.

Aspect	Benefits
Features and Functionality	User functionality and features available in Intelligence Server become accessible to client software automatically. Enabling new features within Intelligence Server automatically makes them available to the correct users.
Project Development and Business Definitions Modification	Project development and business definition modifications at the Intelligence Server are reflected in the client software immediately. Creating new business definitions, and modifying objects at the Intelligence Server automatically rolls them out to users.
Administration	Changing administrative settings in Intelligence Server automatically applies them to all client software. Modified user security permissions and privileges are automatically applied to all client software.
Security	Connections to any secured data source are only established by Intelligence Server, allowing for higher ease of administration and a more secure infrastructure.

Figure 5.2 The centralized metadata repository provides many benefits across the enterprise.

5.2 Intelligence Server Engines and Query Flow

Although Intelligence Server is installed and administered as a single unit, it is built up from a group of federated, well-defined, modular, and reusable engines coordinated by a management engine. All the engines within Intelligence Server are multi-threaded for maximum throughput, and linked by a highly optimized communication bus. The main engines of MicroStrategy Intelligence Server are:

- Security Engine – manages user authentication and authorization
- Object Browsing Engine – presents the folder navigation of the BI applications
- Prompting Engine – manages the presentation and resolution of prompts
- Resolution Engine – interprets business definitions for efficient execution
- Command Engine – manages all engine capacities and job queues.
- Cache Management Engine – retrieves report data from caches and history lists
- Dynamic SQL Engine – generates optimized SQL for relational data sources
- Dynamic MDX Engine – generates optimized MDX for multi-dimensional cubes
- Freeform SQL Engine -- bridges business definitions with relational data sources
- Query Engine – directs data retrieval from the data sources
- Analytical Engine – performs additional calculations, cross-tabbing, and subtotaling of results
- Formatting Engine – adds rich formatting and converts results to the required output format
- Monitoring Engine – displays and logs current Intelligence Server activity
- Scheduling Engine – gathers and runs reports attached to time and event schedules

The query flow described below follows a typical job request to run a standard report. To respond to this query, Intelligence Server dynamically processes multiple tasks with coordination provided by the Command Engine.

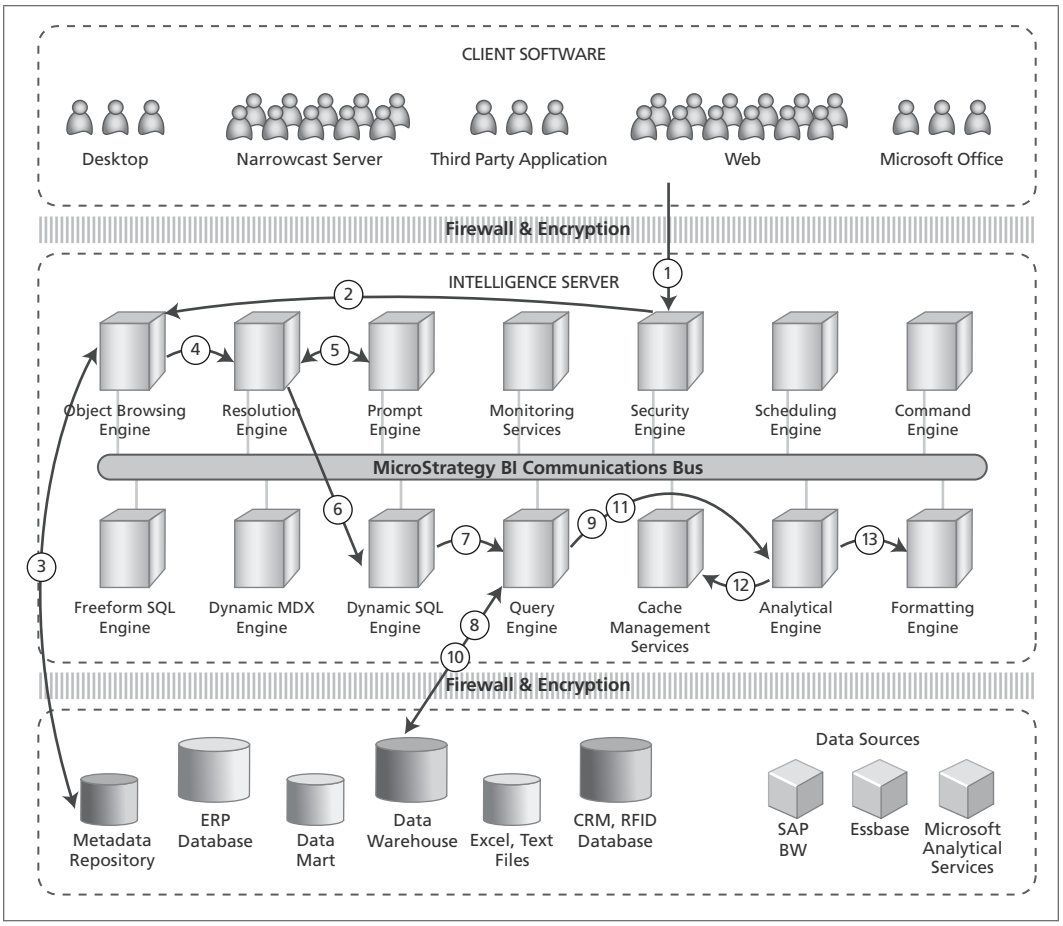


Figure 5.3 Schematic of a query through MicroStrategy Intelligence Server.

1. MicroStrategy Intelligence Server receives a logon request from any interface. The Security Engine authenticates the user, and allocates the appropriate authorization. If third-party authentication is used, the Security Engine receives authentication credentials from the third-party before providing the proper authorization information.
2. The user navigates through the available applications, folders, and business definitions to run a report. This activity is serviced by the Object Browsing Engine. The information is retrieved from the metadata repository or from caches via the Cache Management Engine.
3. If the metadata information is not cached, a connection is made to the metadata repository to retrieve the required object definitions. These object definitions are cached in memory, and managed by the Cache Management Engine.
4. When the user runs a report, the information is handed over to the Resolution Engine for analysis. If the Cache Management Engine determines that a valid cache exists for this report, the cached result set is passed to the Analytic Engine and Formatting Engine before the final formatted report is sent to the user.
5. If prompts are included, the Prompting Engine manages the user input, resolves the prompt answers, and hands the information back to the Resolution Engine. Possible prompt answers are retrieved from cache if available. Otherwise, a connection (not shown) to the data source is created via the Query Engine.
6. Once all required information is gathered for query generation, the Dynamic SQL Engine generates the optimized SQL for the specific data source. For multi-dimensional cube data sources, the Dynamic MDX generates optimized MDX.
7. The Dynamic SQL Engine hands the SQL to the Query Engine for execution against the data source.
8. The Query Engine connects to the relevant data source, and runs the query. If a valid connection is cached, that

connection is used to reduce overhead. Upon completion, the final report results are retrieved.

9. If intermediate calculations are required that are not supported by the data source, intermediate data is transferred back to the Query Engine, and handed over to Analytical Engine for processing.
10. The intermediate results are returned to the data source to complete the query. Steps 9 and 10 can be repeated many times if necessary.
11. The report results are handed over to the Analytical Engine, which performs additional calculations, and applies cross-tabbing, page-by, and aggregation according to the report layout.
12. The final result set is added to the report cache, and indexed by the Cache Management Engine. Other nodes in a clustered environment are notified that this report cache is available. This result set is now available for any matching requests from any user interface.
13. The Formatting Engine applies all the report formats – fonts, colors, images, thresholds, alignment, size, bands, and graphs – and converts the data into the required output type – XML, PDF, Excel, CSV, and HTML. The XML is cached and available for all other Web users.
14. The final formatted report is delivered to the user.

Throughout this process, the Command Engine allocates resources between different queries having different priorities among different engines. When multiple users execute multiple reports with multiple priorities, the Command Engine determines how each report will be processed. If a critical report is executed, the Command Engine can decide to suspend the execution of lower priority reports to service the critical report. In addition, when a particular engine comes under relatively heavier load, the Command Engine shifts processing threads to where they are required.

Engines Maximizes Flexibility, Code Efficiency While Maintaining Control

By bundling key processing logic into individual engines, Intelligence Server gains added agility to provide best-of-breed performance and data integrity. The benefits of this architecture include:

- Flexibility
- Code Efficiency
- Scalability
- Consistent Data
- Improved Monitoring

Resources available to these engines can be increased automatically to maintain optimum performance under changing system loads. Different engines provide the flexibility to interrupt processing to expedite higher priority requests. Engines are reused and revisited improving the efficiency of the system. With no redundant functionality among engines, consistency is enforced, the same security is always applied, and the same data is always presented.

5.3 Automatic Resource Allocation

MicroStrategy Intelligence Server is a fully multi-threaded application. Each engine is a discrete processing unit in MicroStrategy Intelligence Server, and contains a dynamic number of processing threads to perform its tasks. With multi-threaded processing units, each unit can support multiple users executing different reports asynchronously. In addition, each processing unit can leverage all available processors in a multi-processor system, enhancing the scalability and performance of Intelligence Server.

At startup, Intelligence Server assigns a specific number of processing threads to each engine. After startup, the Command Engine dynamically reallocates threads from one engine to another as the demand for individual engines varies depending on the nature of user activity and reports executed. This adaptive balancing of internal capacity ensures effective use of available resources, and allows the MicroStrategy BI platform to scale while maintaining performance.

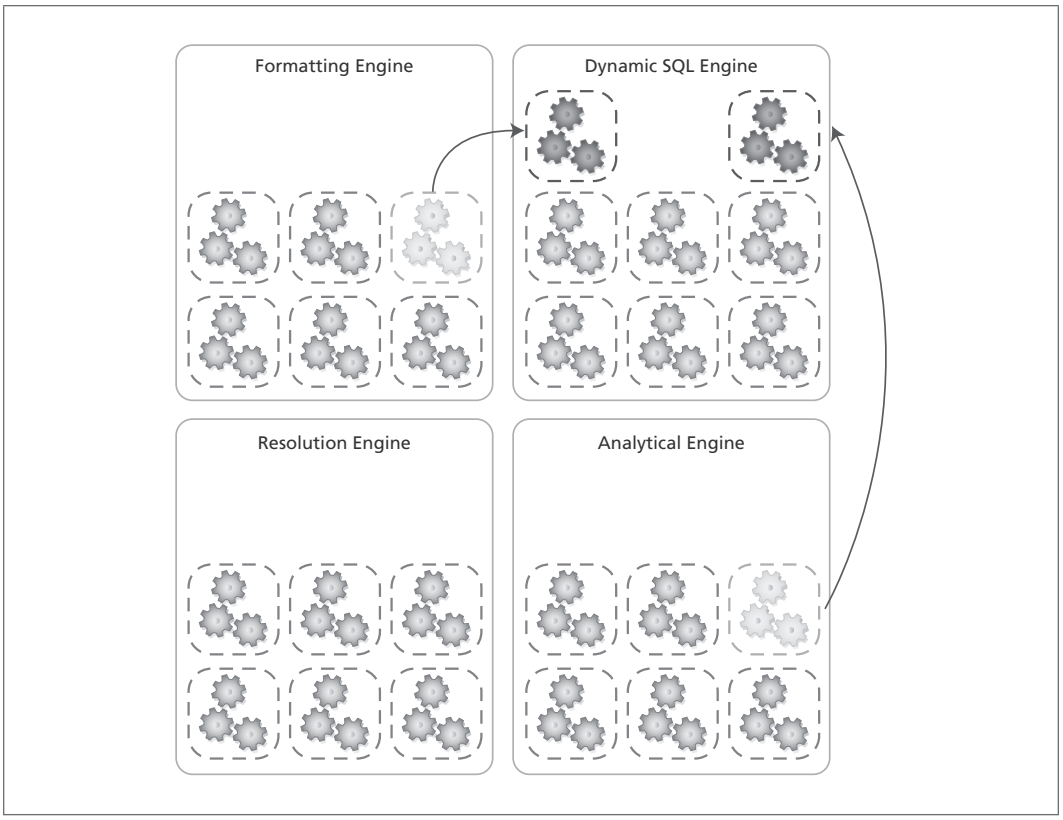


Figure 5.4 Threads are automatically moved to the SQL Engine from other engines to remove a SQL processing bottleneck.

For example, increased demand for reports that contain extensive analytical processing places a higher demand on the Analytical Engine. If the results for most of these reports are cached, the Dynamic SQL Engine will be under a low load. Intelligence Server may reallocate some of the Dynamic SQL Engine threads to the Analytical Engine to process more analytical calculations in parallel. Automatic thread balancing is easily enabled or disabled in MicroStrategy Desktop.

5.4 64-bit Business Intelligence

The amount of memory a business intelligence product can address is determined by the number of addressable bits. The change from 32-bit to 64-bit is a fundamental alteration; for a BI product to be fully 64-bit ready, not only does it need to execute on a 64-bit Operating System, it also needs to port to 64-bits. While other BI products claim 64-bit capable, they may simply execute on 64-bit Operating Systems when in fact the applications are still operating under 32-bit limitations. MicroStrategy Intelligence Server is a fully ported 64-bit BI application supported on 64-bit operating systems.

With fully ported 64-bit Intelligence Server, the capacity of each Intelligence Server instance is increased multi-fold. The benefits include:

- Increased number of users
- Increased cache capacity, and hence improved performance
- Richer BI applications through increased metadata capacity
- Reduced number of physical servers

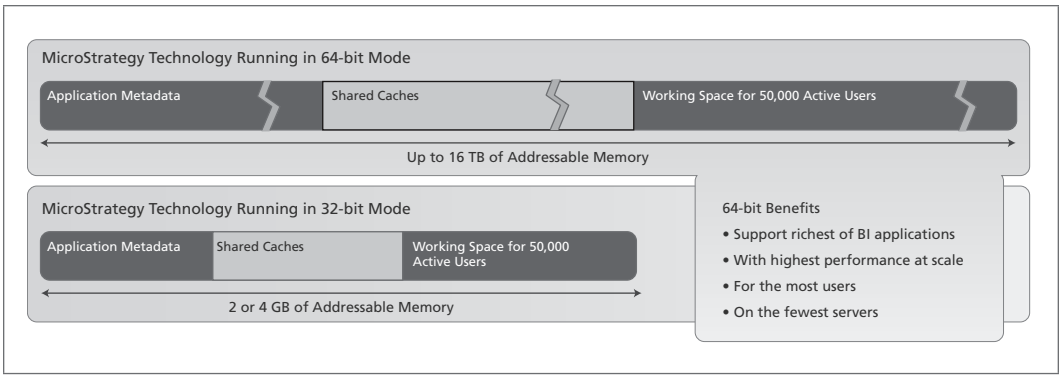


Figure 5.5 Native 64-bit technology in MicroStrategy supercharges BI.

True 64-bit technology increases the amount of data that Intelligence Server can store and access in memory. MicroStrategy 64-bit BI technology seamlessly scales with the growth in users and data sizes while delivering ultra high performance.

5.5 Optimized for Heterogeneous Data Sources

Business needs often dictate that information is retrieved from different data sources. A business intelligence server must be able to communicate with any data source to fulfill these requests. However, different data sources have different connectivity mechanisms, optimization techniques, and result formats. Intelligence Server integrates with all major data sources, is optimized to extract data from these data sources, and can merge the data into a single document.

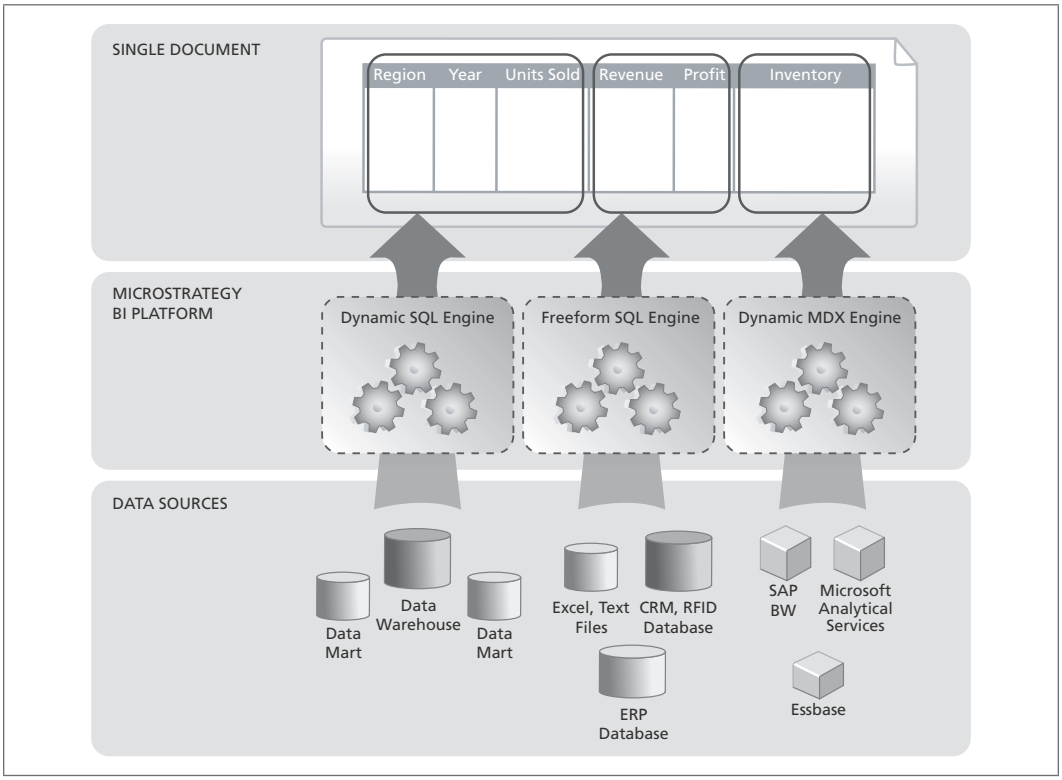


Figure 5.6 Report Services documents contain data from heterogeneous data sources.

Open Database Connectivity (ODBC)

The MicroStrategy BI platform accesses many database systems using a common ODBC layer. MicroStrategy packages wire protocol ODBC drivers which communicate directly to the database through the database's own wire level protocol, eliminating the need for installation of the database's native networking drivers. This approach effectively removes an entire communication layer, further improving system performance and reducing system complexity.

The use of traditional ODBC drivers and the database's native networking drivers requires database client libraries for each version of database accessed by the BI application. By contrast, when using the MicroStrategy BI platform, only one wire protocol ODBC driver per database brand is required. This dramatically reduces administrative overhead and potential conflicts because one wire protocol ODBC driver services all database versions from the same vendor.

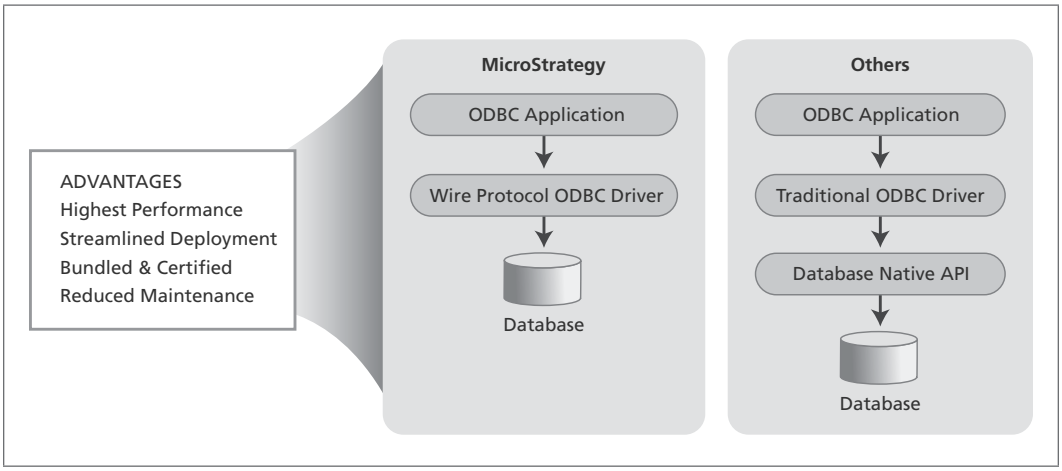


Figure 5.7 Schematic representation of traditional ODBC drivers and wire protocol ODBC drivers.

Dynamic SQL Engine

Database vendors invest heavily in optimizing their Relational Database Management Systems (RDBMS) to keep up with the proliferation of data. Each RDBMS provides many strategies to deal with this trend. The MicroStrategy BI platform, with its Relational OLAP (ROLAP) architecture, complements these strategies. MicroStrategy uses the RDBMS for much of its processing, making the most of the RDBMS investment. No time-consuming and costly additional processing is needed to load the data into a datamart or proprietary cube before users can access data.

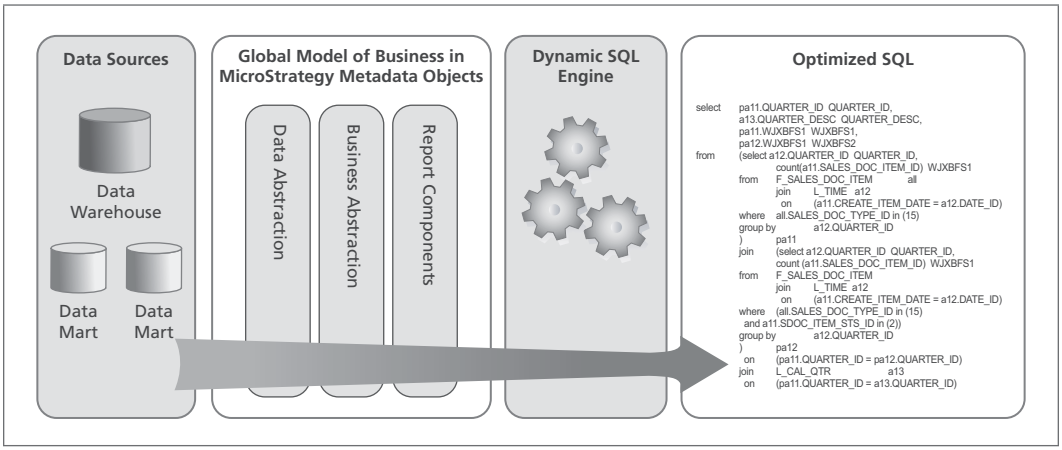


Figure 5.8 MicroStrategy automatically generates multi-pass SQL that is optimized for the specific database being used.

Since each database uses its own optimized Structured Query Language (SQL) syntax, a generic SQL statement performs very differently on different database platforms. The performance difference can be orders of magnitude depending on the complexity of the query. Intelligence Server generates optimized SQL syntax specific to each database platform and version. MicroStrategy Intelligence Server optimizes performance in several ways:

- Multi-pass SQL
- Aggregate Awareness
- Very Large Database (VLDB) properties

Dynamic MDX Engine

MicroStrategy's Dynamic Multidimensional Expressions (MDX) Engine generates optimized MDX syntax for multi-dimensional data sources. Firstly, MicroStrategy connects to SAP using high performance SAP Business Applications Programming Interface (BAPI®) via Java Connectors. For other cube data sources, the connections will use extensible markup language for analysis (XML/A) as a Simple Object Access Protocol (SOAP) interface over Web services.

MicroStrategy's MDX is dynamically generated using the multi-dimensional models imported from SAP InfoCubes, QueryCubes, Operational Data Store (ODS) objects, and from other multi-dimensional cubes. End users can access all the information contained in these data stores directly through the MicroStrategy BI platform. All existing levels, characteristics, hierarchies, key figures, measures, variables, and structures in the cubes can be accessed by end users, ensuring that development efforts are not duplicated. Drilling back into the multi-dimensional cube for more data is automatic and transparent, without any programming or prior design of drill paths.

BAPI takes advantage of all SAP Business Information Warehouse (BW) capabilities, and provides an open, platform-independent interface to access any information that is available through the SAP BW OLAP engine. MicroStrategy is a certified Powered by SAP NetWeaver® and OLAP Business Applications Programming Interface (BAPI) for SAP BW solution.

XML/A uses XML to transmit MDX queries to other multi-dimensional cubes on any platform, allowing access to all data stores that comply with these standards. Using XML/A, the MicroStrategy BI platform fully utilizes OLAP functionality within the multi-dimensional data sources.

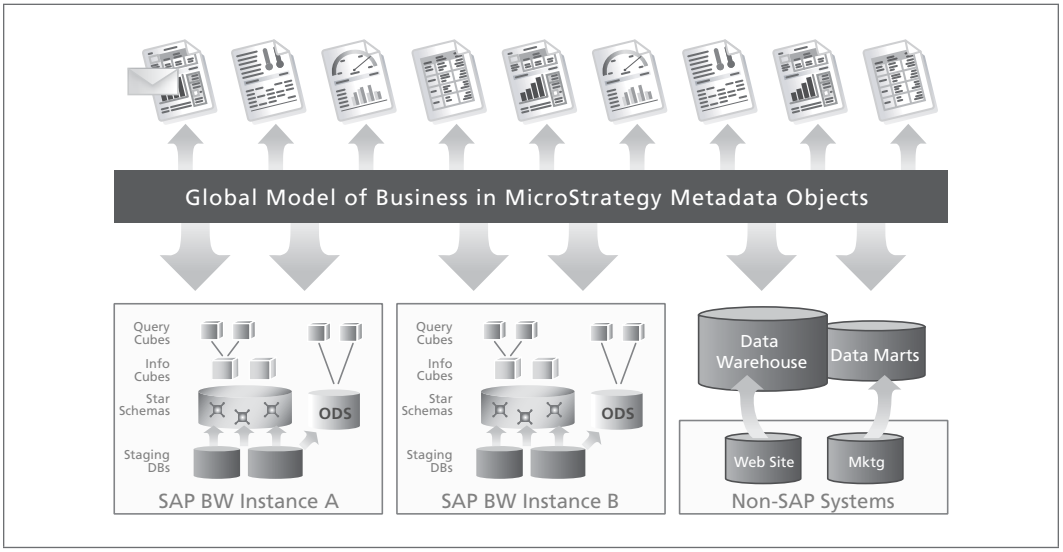


Figure 5.9 Accessing multiple SAP BW Instances together with other data sources.

Freeform SQL Engine

The Freeform SQL Engine in Intelligence Server provides access to data stored in any format accessible by ODBC, including Microsoft Excel files, CSV text files, and other operational databases of systems such as ERP systems, RFID applications, Web Site tracking, Call Center tracking, and CRM systems. It embeds hand-crafted SQL statements directly into the MicroStrategy BI platform to leverage existing investments such as proprietary stored procedures and SQL routines. Freeform SQL statements are subject to the full suite of security features in the MicroStrategy BI platform. Prompts can also be included, increasing the flexibility of the SQL statements.

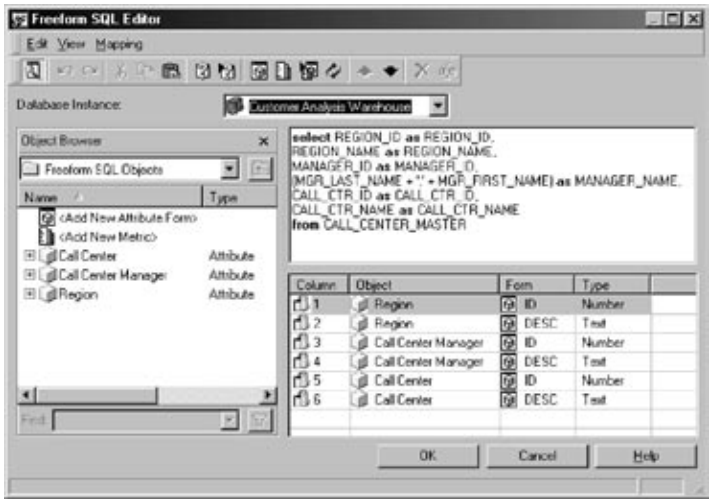


Figure 5.10 Freeform SQL provides the agility to inject prototype data prior to formal inclusion in the data warehouse.

Freeform SQL also provides the agility to rapidly inject information from new data sources into the MicroStrategy BI platform. Prior to committing new processes to extract, transform, and load data from these new sources into the enterprise data warehouse, new data can be retrieved using freeform SQL, and presented to business users through existing reports in an already familiar environment and format.

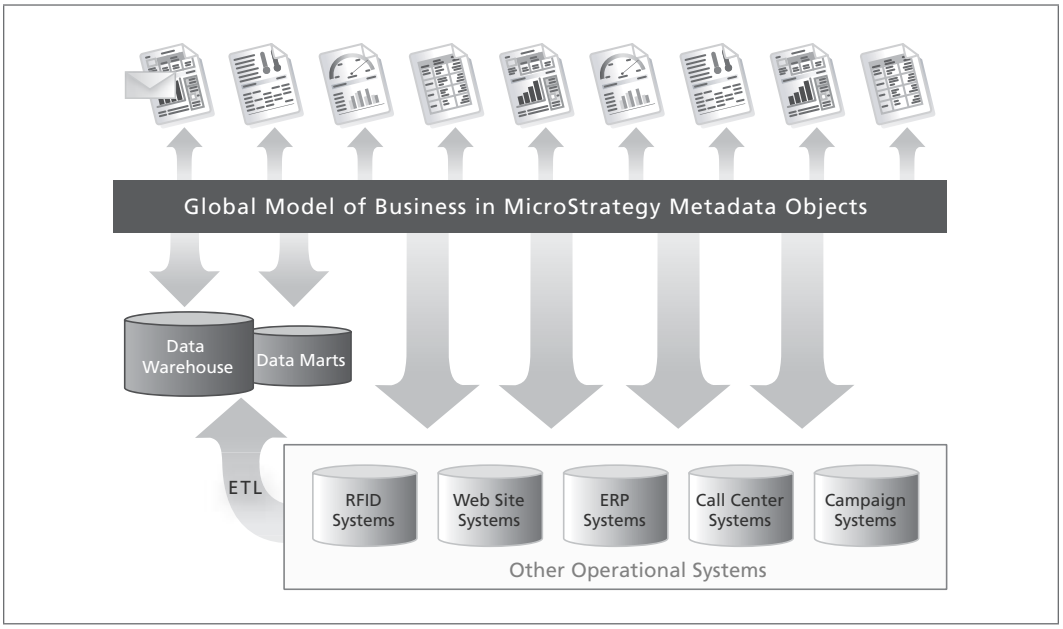
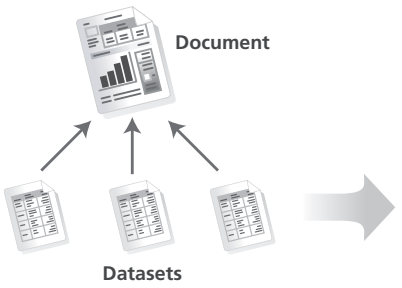


Figure 5.11 Direct access to operational data together with other data sources.

Joining Data from Different Data Sources

Data from each of the above distinct sources can be combined for integrated reporting. After optimization against each different data source, the MicroStrategy BI platform retrieves the data, and then collates and combines it into a unified virtual dataset. When common data exist within the datasets, these are joined within the Intelligence Server. When there is no common data, Intelligence Server builds the relationships internally instead of simply performing a cross join. Unlike other BI products, this approach optimizes memory usage, reduces processing time and enhances performance.



Region	Year	Revenue	Units Sold	Profit
Northeast	2003	$1,230,989	29,368	$298,472
Northeast	2004	$1,103,875	27,380	$266,107
Central	2003	$942,718	22,049	$229,126
Central	2004	$830,552	20,492	$199,603

Figure 5.12 Joining data from distinct data sources into a virtual dataset is automatic and efficient.

5.6 Sophisticated Analysis

It is not sufficient for an enterprise-class BI platform simply to retrieve information from a data source. Business users must be able to apply analytical richness to the data to gain additional insight for accurate, data-driven corporate performance monitoring and decision making. MicroStrategy enables questions, such as how the business is doing, what problems exist, how to solve them, and how to improve operations, to be asked at the enterprise level with a consolidated view. To achieve this, the MicroStrategy BI platform:

- Supports a wide range of analytical functions
- Applies these functions across all data sources in the enterprise
- Supports commonly used analytical techniques
- Performs predictive data mining
- Offers rich presentation methods for easy data consumption

Metrics and key performance indicators available in the MicroStrategy BI platform range from simple aggregations, and counts to advanced financial and statistical models and predictive data mining analysis. With more than 270 analytical functions and operators, MicroStrategy empowers organizations for better monitoring, reporting, and analysis.



Figure 5.13 MicroStrategy analysis can be as sophisticated as required. This example shows the Black-Scholes option pricing model.

Prompts Control the Scope of Analysis

The Prompt Engine interacts with end users to manage all the prompts contained in a report. MicroStrategy prompting allows the user to dynamically control the complete scope of an analysis, not just the filtering criteria. An analysis can be made more comprehensive or more focused by answering simple questions like When, Where, Who, What, and How.

Prompts expand the potential analysis of a single report definition by giving users the flexibility to decide what attributes and metrics to include when the report is run. Users have the flexibility to expand the scope of their analysis to any part of the enterprise data warehouse dynamically, without the need for new reports to be created. Prompts also focus the analytical process by having users provide filtering criteria that limit the cross-section of the enterprise data warehouse to include in the analysis.

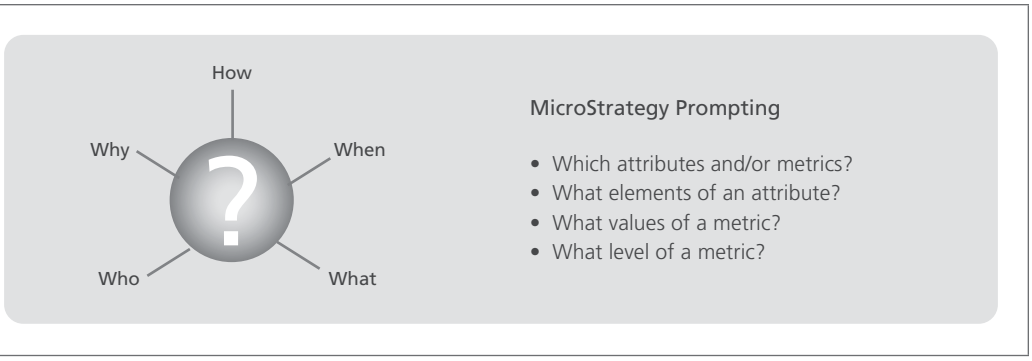


Figure 5.14 Control scope of analysis with MicroStrategy prompting.

Flexible Aggregation Defines the Level of Analysis

An important aspect of analysis is the level at which individual metrics are aggregated. In many cases, specific groups are compared against their entire peer groups. For example, following an advertising campaign targeted at Boston, New York, and Washington, D.C., the marketing analyst can use MicroStrategy to conduct cross strata analysis as follows:

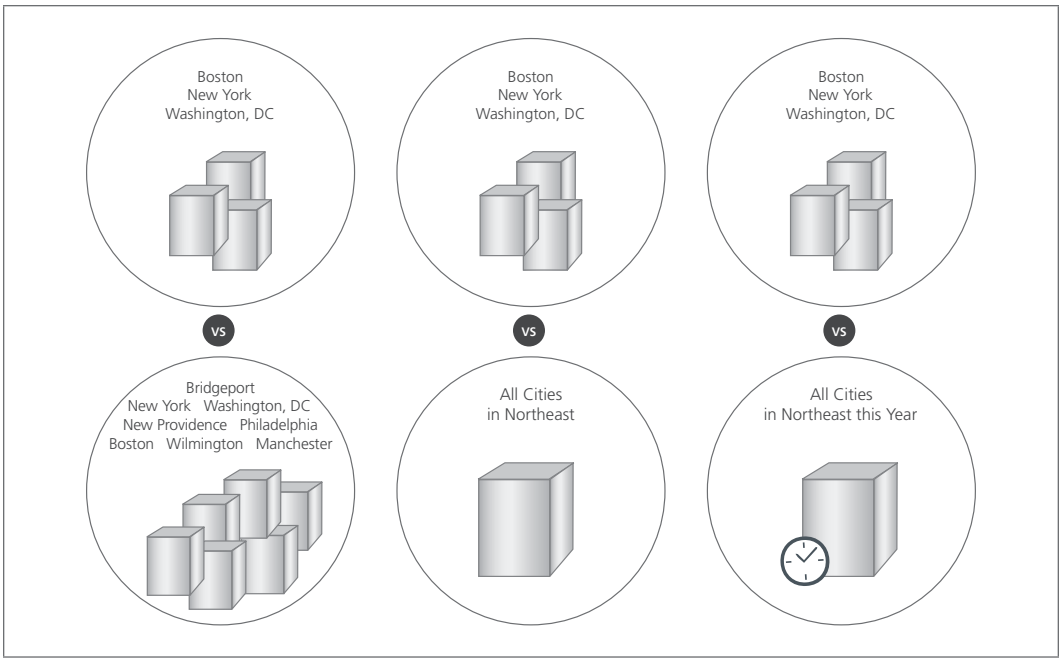


Figure 5.15 Flexible aggregation powers a single report for comparative analysis at different levels.

Each analysis above sheds new light on the impact of the campaign. With other BI products, multiple reports will be defined with complex custom SQL coding to answer each question. MicroStrategy prompts provide users the flexibility to choose, "on the fly", the level at which a specific group should be compared. In all circumstances, data security continues to be enforced.

Additional Enrichment of Data in the Analysis

MicroStrategy Intelligence Server shares analytical processing with the database platform. When the database natively supports a function, it is included directly in the SQL query. If the database platform does not support a function directly, Intelligence Server first attempts to translate the function into a formula that the database can support. If a translation is not possible, the data is retrieved, and passed to the Analytical Engine. The Analytical Engine uses the internal functions to calculate the metric, adds it to the data set, and formats the data before sending the results to the user. Intelligence Server dynamically performs these tasks without any intervention from the administrator or user. This automatic mechanism answers all user questions, regardless of whether the database supports the underlying analytical function.

Business users often need to define their own data groups or clusters of data elements, irrespective of how data is grouped in the database. The data schema within the database cannot always satisfy dynamic business needs. For example, users might want to find the Top 10% of the most profitable customers by different attributes displayed in separate rows on a report. Since this data changes every time new data is loaded into the data warehouse, maintaining this groupings in the database causes significant additional overhead. Intelligence Server solves this problem with Custom Grouping and Consolidations, in which attributes can be grouped into sets that define specific segments without modification to the underlying data. Complex segmentation SQL is automatically generated without end-user coding.

Standard set operations – union, intersection, exclusion, and difference – can be applied across more than one attribute set to refine the segmentation criterion. This is called Set Analysis. For example, in order to isolate a list of target customers for a promotional campaign, sets of attributes can be used to identify customers who have purchased the promotional items or products with a high correlation to the promotional items. Using these multiple sets, a marketing department can isolate target customers, and coordinate supporting activities with stores and suppliers to work towards a successful campaign.

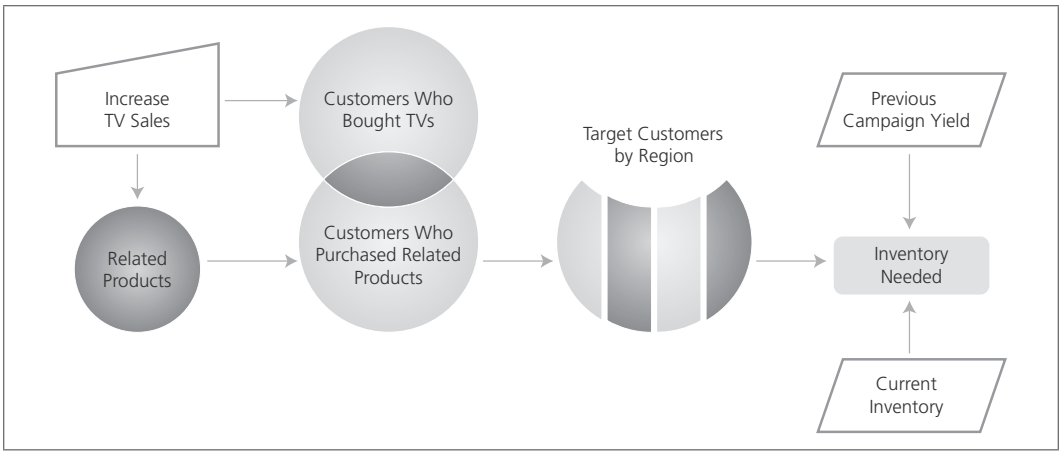


Figure 5.16 Set Analysis to segment a list of target customers for a TV promotion.

Iterative Analysis Technology

Iterative analysis is the dynamic mechanism that represents the continuous interaction and advanced analytical techniques between Intelligence Server and the database. MicroStrategy uses this approach when complex questions cannot be answered using only the database, or when a complex question that requires several steps cannot be answered by a simple SQL query. Not only does the Intelligence Server automatically invoke iterative analysis when required, it also dynamically decides the most optimal location to perform the calculations, based on its own strengths and the database strengths, while minimizing data transfer. The MicroStrategy BI platform is the only BI platform in the industry that performs iterative analysis.

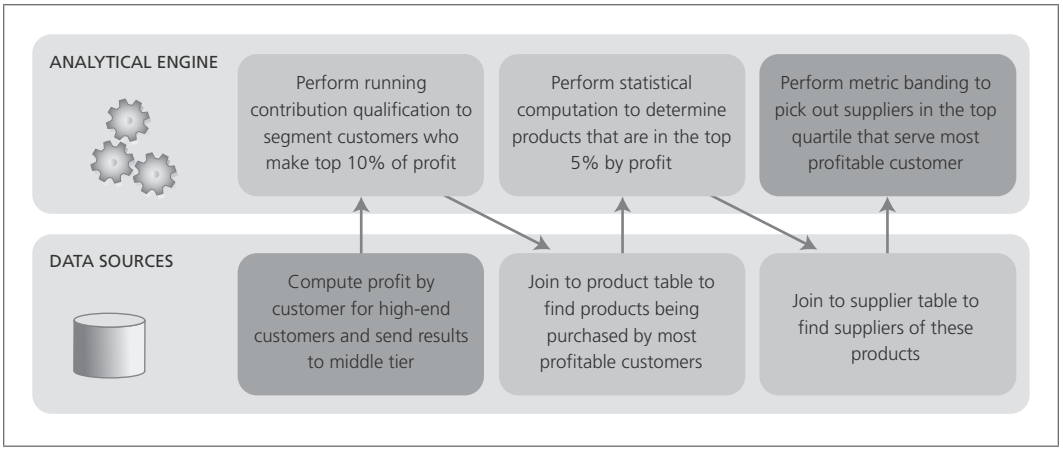


Figure 5.17 Complex iterative analysis is automated and simple without user coding.

Data Mining Integration into Enterprise BI

Data mining is the process of sifting through large amounts of historical data in search of hidden patterns and predictive information using statistical analysis, modeling techniques, machine learning, and database technologies. In other BI architectures, organizations are limited to historic analysis within a Business Intelligence solution, and are forced to perform predictive analysis in a separate system. With MicroStrategy, data mining scoring is fully integrated into mainstream business intelligence. This complete integration elevates the availability of predictive metrics to all decision makers and business users.

The integration of data mining models from other applications is accomplished by importing Predictive Model Markup Language (PMML) into the metadata repository, and automatically creating a predictive metric. PMML is an

XML-based industry standard developed by the Data Mining Group (DMG) to describe predictive models. It supports a number of different data mining algorithms, including Neural Networks, Clustering, Regression, Decision Trees, and Association. MicroStrategy is the first Business Intelligence platform to support the PMML standard. Predictive models are still developed in specialized data mining applications; MicroStrategy manages the data scoring and distribution of the results across the enterprise.

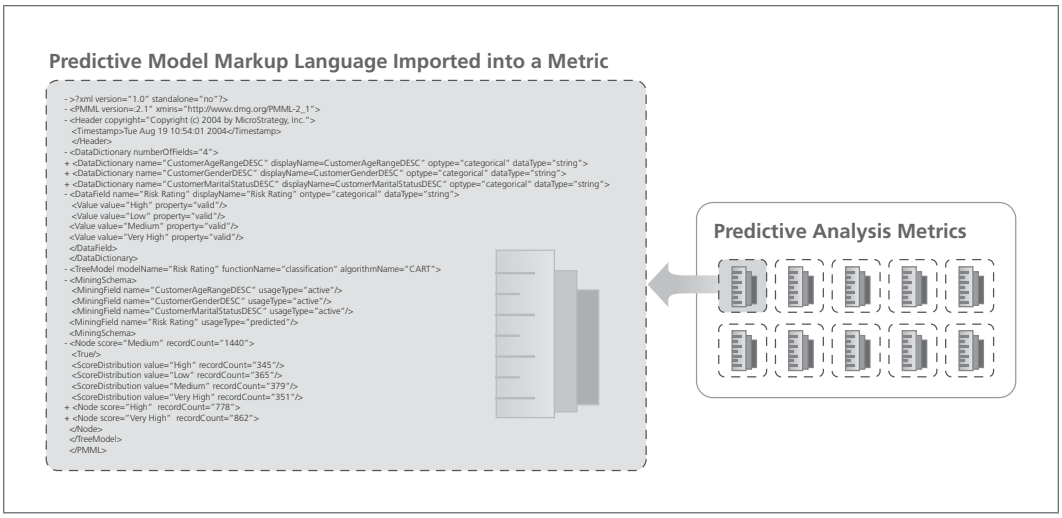


Figure 5.18 Predictive Model Markup Language creates MicroStrategy data mining metrics.

MicroStrategy Contains the Broadest Analytical Capabilities

The following table contains examples of the analytical capabilities available in the MicroStrategy BI platform. BI applications use these capabilities to add analytical richness to enterprise data.

Type of Analysis	MicroStrategy Analytics	Description
Data Summarization	Sum, counts, variance, mean, mode, median	Summarization allows users to have an overview of their business performance. For example, users can view histograms of customer spending by product or age.
Time Series Analysis	Running and moving averages, TY-LY, MTD, YTD, transformations	Time Series Analyses allow users to dynamically compare data from various time periods. For example, users can accurately forecast future purchasing behavior using regression analysis or chart the 12-month moving average of the stock price.
Segmentation Analysis	Custom groups, consolidations, deciling, pareto analysis	Segmentation Analyses permit users to pull together data elements from any dimension in a database, and cluster them as one customized group. For example, enterprises want to segment their customers by many different attributes, such as age, gender or income, so that specific products can be promoted to the appropriate customer segments.
Conditional Analysis	Attribute comparisons, value and percentage ranking	Conditional Analyses permit users to analyze data through a set of "What If" scenarios. For example, users can compare ship and order dates to calculate order fulfillment performance.
Affinity Analysis	Correlation, market basket analysis, attribute affinity	Affinity Analyses allow users to discover hidden relationships between business variables. For example, users may discover that certain products sell better with other products during a specific day of the week.
Hypothesis Testing	Normal distribution, probability, confidence intervals, p-Values	Hypothesis Testing allows users to draw conclusions from sample data. For example, the following questions can be investigated: Do certain colors sell better in different cities? Do female customers respond better to direct mail campaigns?
Predictive Analysis	Neural networks, clustering, regression, trees	Predictive Analysis allows users to discover key business drivers, and analyze their trends for proactive decision making.

Figure 5.19 The MicroStrategy BI platform supports a broad range of analytical applications.

5.7 Cluster-Capable Intelligence Server

In clustering, a group of independent systems, called nodes, work together as a single system. In the MicroStrategy BI platform, multiple Intelligence Servers running on separate machines are clustered to work together as a single logical system. Each MicroStrategy Intelligence Server is cluster-capable out-of-the-box, and requires no additional third-party software. Compared to the clustering architectures used by other BI products, the MicroStrategy Intelligence Server cluster is an all-active peer-to-peer configuration, and is built to minimize resource overhead.

This autonomous configuration minimizes the network traffic by requiring no central management service for clustering. The communication between each node is highly tuned, allowing for maximum information density per communication.

The features of Intelligence Server clustering include:

- Flexible Load Balancing
- Automatic Failover and Fallback
- Dynamic Asymmetric Clustering

Shared Data and Definitions maintain Single Version of Truth

Each node in a MicroStrategy Intelligence Server cluster connects to the same metadata repository as the other nodes, ensuring that BI users, applications, business definitions, security, data, and caches are identical for all nodes in the cluster. Report results are always the same regardless of which node executes the report. Even when new or updated definitions are added to a node, any changes are immediately available to all nodes within the cluster. This ensures data integrity, and provides a single version of the truth.

Adding nodes to a single Intelligence Server deployment or to an existing cluster requires minimal administrative effort. Aside from simply ensuring that each Intelligence Server node can access the other nodes via the network, a new node simply needs to load the same Server Definition in the MicroStrategy Metadata Repository. Modifying cluster membership is easy and straightforward using either MicroStrategy Desktop or MicroStrategy Administrator's Command Manager.

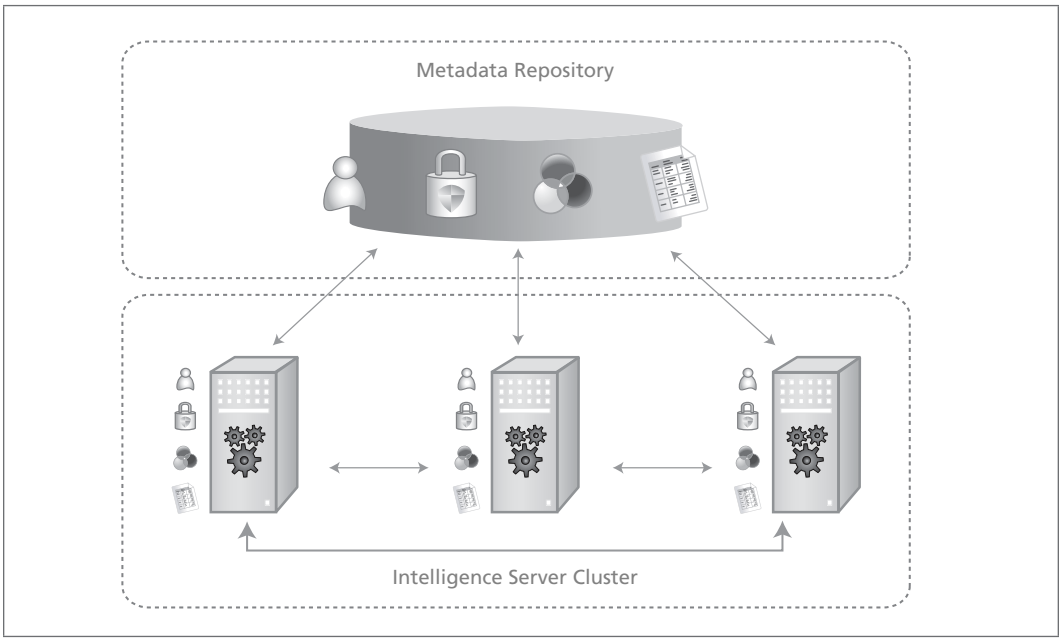


Figure 5.20 All nodes share the same security, business definitions, data, caches, and users, ensuring a single version of the truth for all business terms and measures.

5.8 Multi-Level Shared Caching

Caching is the process of temporarily storing data in the BI platform in memory or on disk, closer to users, to reduce the processing time of job requests. Rather than going to the database for each query, a BI system can leverage cached data of previous job requests for subsequent requests. These job requests include browsing metadata, providing prompt answers, and running or creating reports. As the number of business intelligence users increases, the chances that two users will request the same information increases. As more users enter the system and create more caches, average system response time decreases.

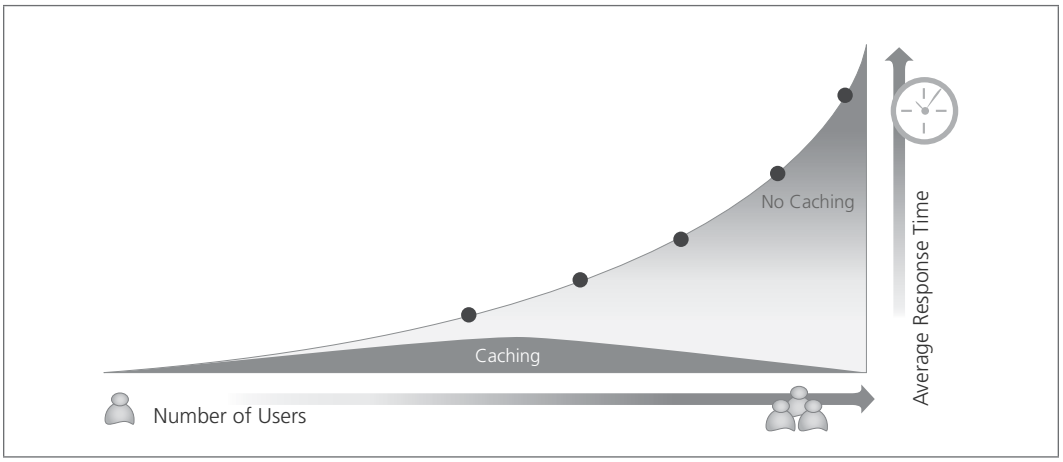


Figure 5.21 Caching dramatically reduces average response time.

While other BI products may simply provide report caching, the MicroStrategy BI platform implements multi-level shared caching. At each strategic point in a user interaction, key data is cached to boost performance. The result is a significantly improved end-user experience and best-of-breed BI performance. The types of MicroStrategy BI platform caching are:

- Metadata Object Caching
- Attribute Element Caching
- Database Connection Caching
- Report and Report XML Caching

In addition, within a clustered Intelligence Server configuration, caches are synchronized across clustered nodes without massive data transfer between nodes, and users connected to one Intelligence Server node benefit from the caches stored by another Intelligence Server node.

Report Cache Creation and Delivery

Caching is only effective if the correct data is served. As a part of the MicroStrategy BI platform's integrated security, Intelligence Server ensures that only the right cache is served to the right users. MicroStrategy uses a sophisticated cache-matching algorithm that matches not only the components of caches or Intelligent Cubes, but also the versions of those components and the user's security context. Examples of information that identifies a cache include:

- Report ID or Intelligent Cube ID
- Report version ID or Intelligent Cube version ID
- Template ID

- Template version ID
- Filter ID
- Filter version ID
- Security filter ID
- Prompt answers

Upon receiving a report request, Intelligence Server verifies that the user is sufficiently authorized through privileges, permissions and the access control list of the objects requested. If caching is enabled for the report request, Intelligence Server proceeds to verify if there is a valid cache for the request using a well tuned hash table. The use of a hash table compresses all subsequent logical verification steps into a single fast and accurate step.

The details of the logical steps are shown in the following diagram. Only after a number of verification checkpoints, including verifying the security context, are cached data returned to the user. The security context includes the user's security filter, user's database login, and the user's connection mapping. If a cache is not available, Intelligence Server automatically generates a highly optimized query against the data source on behalf of the user. This implementation dramatically speeds up cache matching and performance.

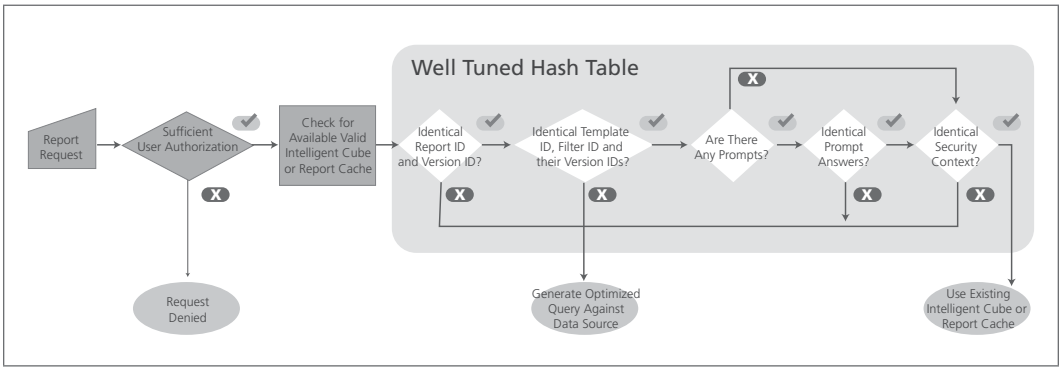


Figure 5.22 Intelligence Server cache matching algorithm ensures correct caches are used.

Cache Management is Automatic and Transparent

MicroStrategy Intelligence Server automatically optimizes and maintains all caches. The Cache Management Engine in Intelligence Server automatically tunes in-memory cached data to meet user, and manages the amount of memory resources used for caches. When this exceeds the configurable value, caches are off loaded onto disk using a Least Recently Used policy. In this manner, the highest number of most frequently used caches are maintained in memory. Compared to on-disk cached data, in-memory cached data provides a much higher degree of performance boost, and Intelligence Server ensures that the most popular cached data remain in memory for maximum benefit.

Caches are automatically updated whenever any underlying object of a particular cache is changed. For example, if the definition of a metric is altered, the new definition is propagated to all reports and other objects using that metric, and all caches affected by the change are automatically invalidated. These caches are automatically recreated the next time the reports or prompts are run.

When Intelligence Servers are clustered together, every cache created by any node in the cluster is available for all clustered machines. The cache hash lookup table on all the nodes is automatically synchronized, making every cache available to users on all nodes.

Intelligence Server statistics provide information on the most frequently used reports, reports with the largest result sizes and reports with the longest query execution times. Caches for these reports can be created during off-peak hours using Intelligence Server scheduling. Administrators can schedule reports to be cached based on time or event triggers. For example, whenever new data is loaded into the enterprise data warehouse, an event can be triggered for Intelligence Server to update related caches.

As the number of reports in a BI application grows, ensuring that the latest data is cached and used becomes more difficult. Some BI products simply purge all cached data, and rebuild all caches from scratch when more data is added to the data warehouse. MicroStrategy Intelligence Server provides a means to integrate with different data loading processes, especially if they update different tables. Only cache files that use these tables are purged and recreated, while the remaining caches are left intact. This maximizes the utilization of existing caches by minimizing redundant cache updates.

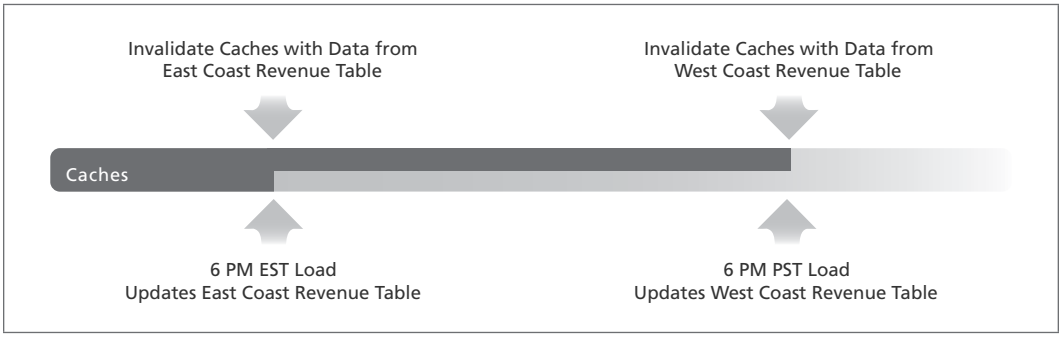


Figure 5.23 Table-based cache invalidation synchronizes caches with data sources.

ROLAP complementing Intelligent Cubes

Most BI vendors use custom-made proprietary cube databases that are complicated to build and only hold relatively small data volumes for cube analysis. As the demand for more BI applications, more users and more sophisticated analysis grows, hundreds, even thousands, of overlapping cubes are needed to cover all combinations of data subsets, summarization levels, and security profiles for different users across multiple applications. These ever-growing collections of cubes, or "cube farms", create an immense burden on IT departments that generate the cubes, pre-calculate the summarizations, distribute them to users, and retire them when their data become outdated.

By contrast, MicroStrategy uses a ROLAP architecture that models the structure of the data sources as relational multidimensional structures inside the metadata repository. This approach provides the same standard OLAP functionality found in cube-based solutions – cross-tab, page-by, pivot, sort, filter, and drill – but the entire breadth and depth of the data warehouse is available for analysis with no complex steps needed by IT. No further extraction of data into a limited and proprietary structure is needed. The previous tradeoff for a ROLAP-only approach (somewhat slower response-times due to the vast amount of data that can be processed) is eliminated with Intelligent Cubes.

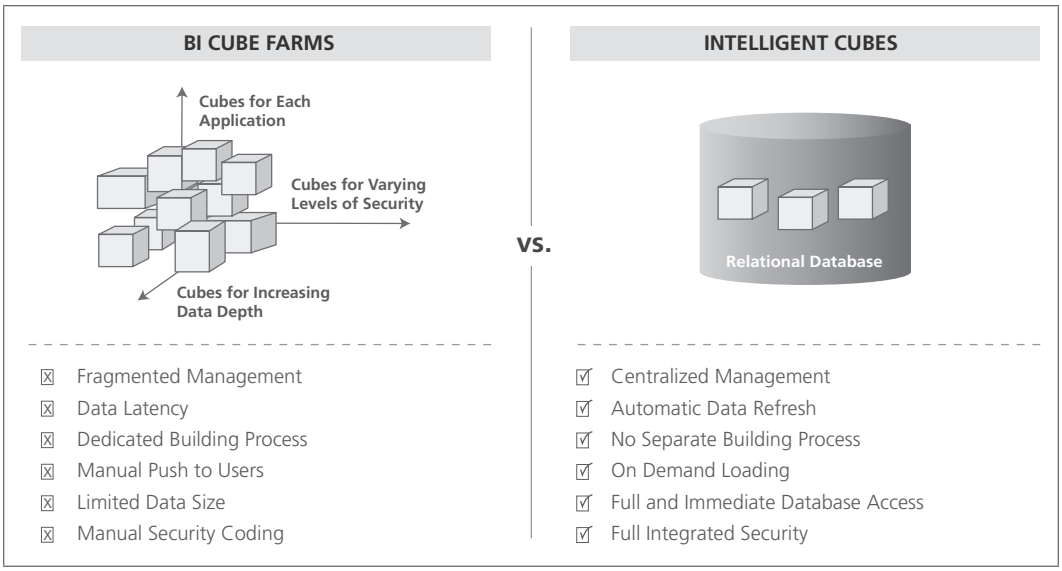


Figure 5.24 Intelligent Cubes enhance performance while providing break-through access.

No Separate Cube Creation Process

IT administrators of other BI software products spend most of their time managing and updating cubes in various locations, such as user desktops, Web servers, and file servers. Cube data is duplicated across all these locations, and this quickly becomes impossible to manage and keep synchronized with the latest data. MicroStrategy OLAP Services delivers fast OLAP by automatically creating and populating Intelligent Cubes in report cache files for use across the entire enterprise.

Intelligent Cubes are designed by the same end-user report creation process. Designers place related business definitions together, insert any required analytics, and scope out the dataset using filtering. New Intelligent Cubes are either manually created by simply executing the source report or automatically when the report is scheduled. Once created, the Intelligence Cubes are loaded into memory automatically when needed and discarded when not.

Intelligent Cube definitions are stored in the centralized metadata, and are automatically created and removed – with cube data always up-to-date and synchronized – without an administrator's involvement. Given that Intelligent Cubes are part of the entire cache refresh system, refreshing Intelligent Cubes can be made an automatic part of the Data Load process. Updating Intelligent Cubes is seamless, injecting flexibility, simplicity, and speed into the MicroStrategy BI platform, and leading to a high performance analysis environment.

Speed-of-Thought Analysis

Intelligent Cubes access data stored directly in memory in Intelligence Server. Full analytical reporting is performed without accessing the data warehouse with every request. This provides a dramatic boost in performance, as the responses to data manipulation occur almost immediately.

Even though the dataset within the cached subset is bounded by the report definition, users retain access to the entire data warehouse when needed. Unlike other BI products, MicroStrategy OLAP Services automatically detects queries outside of the cached subset, and generates a highly optimized query against the specific data source to retrieve the additional data. This process works behind the scenes, and is seamless to the end-user. From the end-user's perspective, accessing data from an Intelligent Cube, from a report cache, or reaching through to the data source is transparent. With Intelligent Cubes from OLAP Services, the MicroStrategy BI platform embeds Speed-of-Thought analysis.

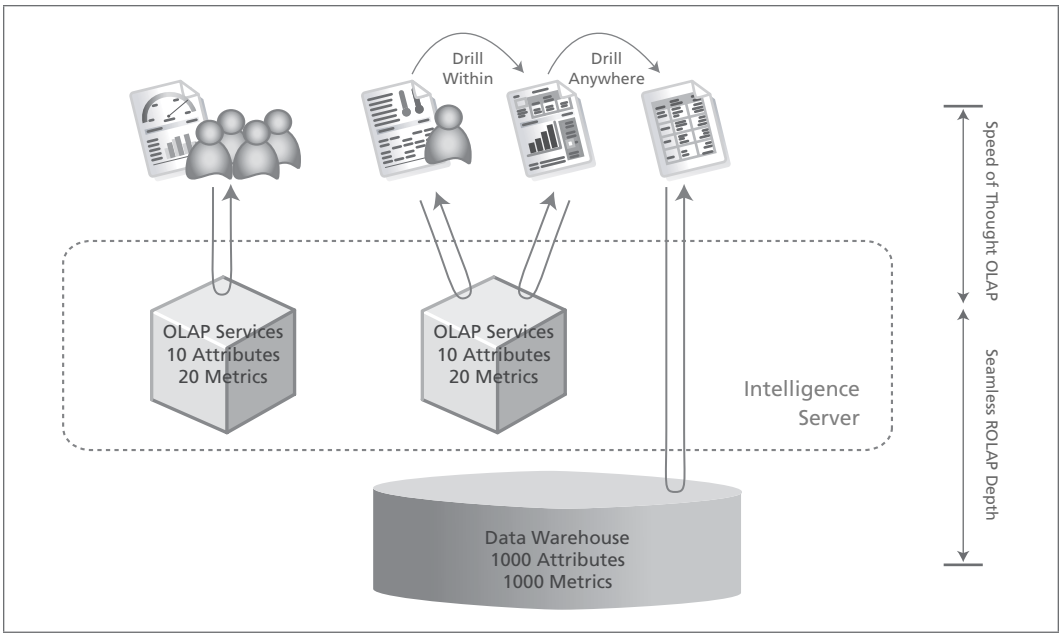


Figure 5.25 Drilling within and outside of Intelligent Cubes is seamless, embedding speed-of-thought analysis.

The following report manipulations are extremely fast because they are executed directly against Intelligent Cubes, without generating additional SQL queries:

- Drilling
- Pivoting
- Page-by
- Sorting
- Subtotals
- Adding derived metrics and calculations
- Modifying column labels (column aliasing)
- Adding/removing information from the report view
- Filtering report cache results
- Formatting the report view
- Outline mode
- Banding
- Reset data to original report

5.9 Scheduling Reports and Administrative Tasks

Scheduling is the process in which tasks are automatically executed at a specific time or after an event occurs. The MicroStrategy BI platform is designed for both scenarios. This integrates the MicroStrategy BI platform into existing business processes, and automates tasks within Intelligence Server. While other BI products only provide for the scheduling of report executions, the Intelligence Server also enables the scheduling of administrative tasks. The following tasks are available for scheduling within Intelligence Server:

- Report Execution
- Cache Management
- History List Message Management
- Project Status Management

Frequently updated reports can be scheduled to automatically execute after each new data load or at specified times. As part of a regularly timed data load, Intelligence Server automatically executes reports at specified times. In scenarios where the data load process is less predictable, the data load process can fire off an event which automatically triggers report executions within Intelligence Server. Similarly, various administrative tasks can be scheduled. For example, when business usage cycles are demarked by time or by event, Intelligence Server can automatically off-load or load projects, invalidate caches or delete user History List messages.

5.10 Exporting Data

While other BI products provide limited options to export data, the MicroStrategy BI platform offers numerous options. Within the MicroStrategy BI platform, users can export their report results into various formats:

- HTML
- Microsoft Excel (No formatting, with formatting, or CSV files)
- PDF file
- Microsoft Word
- Microsoft Access
- Text file (Comma, Tab, Semi-colon, or Space delimited)

These different formats are useful for different purposes. For example, PDF documents are often used for presentation quality data summaries suitable for executive review. Excel documents are useful for offline data analyses, and are also strongly preferred by finance departments who are skilled at using Microsoft Excel. Using sophisticated analysis within the MicroStrategy BI platform, users can build data subsets for users in other systems. Microsoft Access format, delimited data formats or datamarts are often used to handle large data sets such as address lists where a customized subset of the data is needed for use in third-party applications, such as a campaign management tool.

When exporting data into Microsoft Excel workbooks, reports can be associated with the MicroStrategy BI platform using MicroStrategy Office; when underlying data is updated, the report results in the Microsoft Excel workbooks can be updated. In addition, with graph reports, grid data can be exported in addition to the graph representation. This functionality is called "Live Charts". By the inclusion of the raw grid data within Microsoft Excel, a dynamic workbook is generated, giving users opportunities to further analyze the data while offline.

Datamarts

With access to the entire data warehouse along with sophisticated analysis from the MicroStrategy BI platform, users are often keen to use analysis results in other applications within the enterprise. When result sets are large, writing the results back into database tables as datamarts provides a convenient path to making the data accessible to different applications. To optimize this process, the Intelligence Server utilizes parameterized queries, and provides a list of user configurable options. This flexibility allows for maximum customization to any specific database environment, ensuring maximum performance. As datamart reports can also be scheduled just like any other report, this workflow can be triggered along with data loads.

```
CREATE [Table Qualifier] TABLE [Table Descriptor][Table Prefix] <table_name>
 [Table Option] ( <column_expressions> ) [Table Space][Create Post String]
```

Figure 5.26 Customizable parameters within [ ] brackets provide maximum flexibility and performance.

5.11 Formatting Data

Once all report data processing is complete, Intelligence Server formats the data using a highly sophisticated formatting engine. Each cell within the result data is formatted accordingly. The following are available cell formatting options within the MicroStrategy BI platform:

- Number Format – General, Fixed, Currency, Date, Time, Percent, Fraction, Scientific, Special, Zip Code, Telephone, Custom
- Alignment – Text Alignment: Vertical or Horizontal, Wrap Control
- Font – Type, Language Script (Character Sets), Size, Bold, Italic, Strikeout, Underline, Color
- Border – Line Style, Location: Top, Bottom, Left, Right, Color
- Background – Fill Color, Pattern Color, Pattern

As each cell can represent data at an intersection of business definitions, the formatting engine combines information from multiple business definitions to create the final report presentation. The final format of a single cell is built from five different formatting layers, with the lowest level overriding higher levels. The levels are listed below, with the highest level listed first:

1. Pre-defined formats in Autostyles
2. Metric Formatting
3. Report Banding Formatting
4. Custom Group Element / Consolidation Element Formatting
5. Report Thresholds Formatting

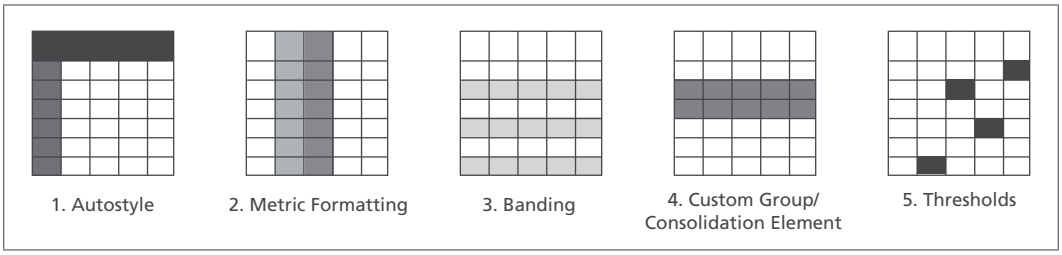


Figure 5.27 Five layers of formatting definition in MicroStrategy grids.

An Autostyle is a set of pre-defined format styles provided by the MicroStrategy BI platform. It consists of a collection of formatting definitions for different sections in a report. These include row-level headers, row-level values, column-level headers, column-level values, metric headers, metric values, subtotal headers, and subtotal values. Within the MicroStrategy BI platform, users are able to create their own custom-defined Autostyles. The other formatting layers are controlled by formatting information within specific business definitions such as metric, custom groups, consolidation, and reports. These 5 layers of formatting information enable the MicroStrategy BI platform to provide highly sophisticated formatting for final report presentation within MicroStrategy Desktop, MicroStrategy Web using XML or any of the export formats listed in the exporting section above.

5.12 Simplified Central Administration

The administration of the MicroStrategy BI platform is centrally located at the Intelligence Server. Unlike other BI platforms, the Intelligence Server components and services were built organically. As a result, there is a single unified administration interface within MicroStrategy Desktop, and full-featured, integrated scripting support from

MicroStrategy Administrator – Command Manager. The administration of Intelligence Server is comprised of two main components:

- Monitor
- Control

Together, these two components provide full oversight into the MicroStrategy BI platform.

Monitoring Gives Instant insight

MicroStrategy Intelligence Server provides real time system information in all aspects of the platform. From MicroStrategy Desktop, various information feeds such as the following are available:

- Jobs
- Projects
- Users
- Database Connections
- Scheduled Jobs
- Caches
- Clusters

Each of these monitors provides details on the number of items and a snapshot of their statuses. For example: The Job Monitor below shows 4 jobs currently executing. The user executing the report "Top 10 Profitable Products" is currently providing prompt answers. A document called "Customer Analysis Scorecard" is currently executing with two child reports named "Q3 Top 10 Most Profitable Products" and "Q3 Top 10 Customers by Profit". All of these reports are executed by the user "Administrator" connected to the project "Customer Analysis Module". The reports are submitted within a short time of each other and from the same client machine.

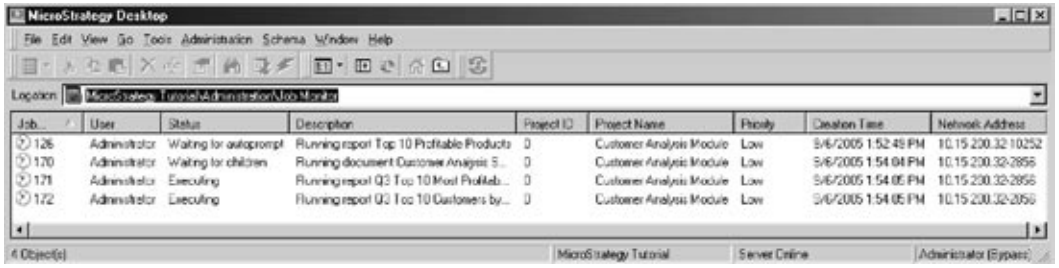


Figure 5.28 The Job Monitor provides real time details on report executions.

Similarly, another example is the Cache Monitor. This monitor provides real time insight into what caches are available and their states. Below, there are 5 caches. The cache monitor reflects the projects in which their reports reside, the cache sizes, statuses, and creation times, and details on how many times each cache was used.

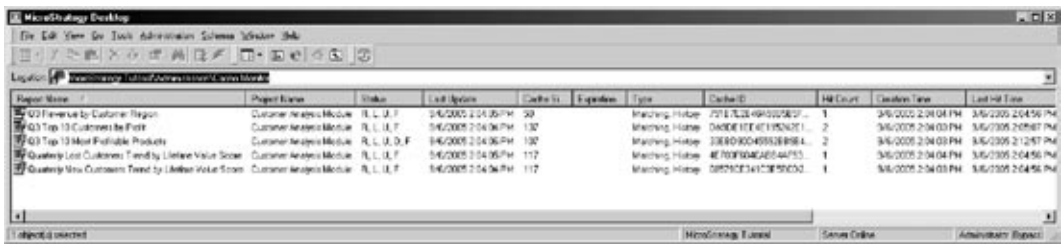


Figure 5.29 The Cache Monitor provides real time details on available caches.

The status of each cache is also listed in the cache monitor. This provides additional insight into how the caches can be used. The following is a list of cache statuses:

- Ready: Cache can be used.
- Processing: Cache is being created.
- Invalid: Cache cannot be used because the report definition has changed.
- Expired: Cache cannot be used because it is too old.
- Filed: Cache is written to disk.
- Loaded: Cache is in memory.
- Dirty: Cache in memory is newer than cache on disk.
- Updated: Cache has been updated.

In addition to real time data, Intelligence Server also logs historical usage statistics, allowing comprehensive analysis of system usage.

Take Action With Controls for Maximum Efficiency

By default, Intelligence Server is already preconfigured, however, it is also designed with various levers and controls at numerous levels and components, allowing the BI platform to be configured and optimized for any deployment scenario. Complementing the extensive amount of monitoring information available, Intelligence Server provides full control into the business intelligence system to maximize the system's efficiency. Control of the Intelligence Server can be broken down into three main categories: Server Workload, Memory and Prioritization.

Administrators can control the following in order to govern the server workload:

- Number of Network Threads
- Maximum number of executing jobs
- Maximum number of user sessions and per project
- Maximum number of jobs per project, user, or session.
- Number of database connections
- User idle time
- SQL cancel
- Maximum report execution time
- Maximum recent reports for manipulation
- Project Failover Latency
- Configuration Recovery Latency

Memory management is a key requirement for an enterprise class platform. MicroStrategy Intelligence Server governs memory usage through the following parameters:

- Report execution size
- Number of rows returned
- Number of elements returned
- Maximum number of XML cells
- Maximum number of XML drill paths
- Maximum memory consumption for XML, PDF, and Excel
- Maximum RAM usage for object cache
- Maximum RAM usage for report cache
- Maximum number of caches

- Maximum RAM usage for element cache
- Maximum RAM usage for Intelligence Server process
- Minimum reserved RAM for external processes
- Minimum machine free RAM
- Maximum Intelligence Server use of virtual address space
- Maximum Single Allocation Size
- Memory Request Idle Time
- Maximum RAM usage for Working Set
- Cache Lookup Cleanup Frequency
- Maximum number of messages per user
- Maximum message lifetime
- Maximum Attribute Elements to Display

Prioritization of report requests is critical to ensure that users receive the level of service appropriate to their positions. Without this capability, a CEO would need to wait at the end of the queue before his or her requests are processed. To avoid such scenarios, all requests must be prioritized, and routed through a central server. The Intelligence Server prioritizes jobs based on:

- Report cost
- User groups
- Projects
- Request type
- Application type

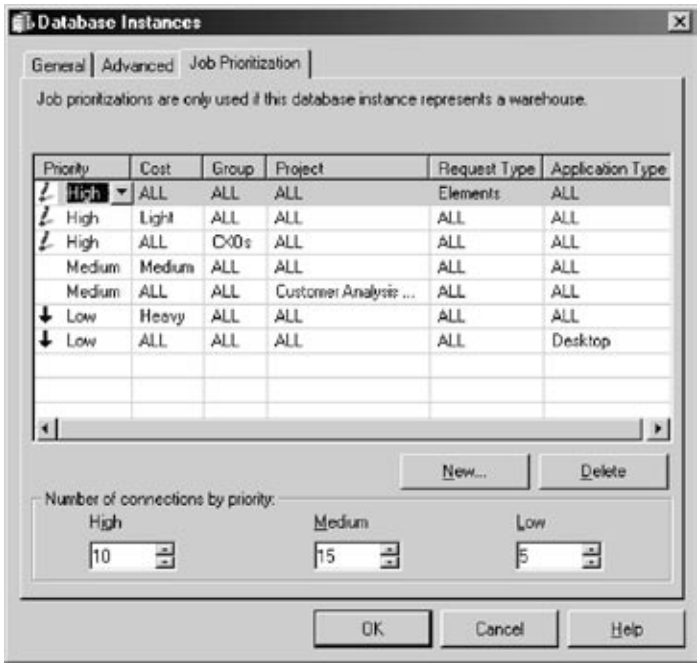


Figure 5.30 Parameterized processing threads provide flexible report request prioritization.

By combining the wealth of information available via monitoring and near real time statistics, together with the capability of scripting control actions, MicroStrategy Intelligence Server provides the foundation for sophisticated system administration.

5.13 Summary

MicroStrategy Intelligence Server provides enterprises with one central location for all user querying, reporting, and advanced analysis needs. With optimized query generation for different data sources, multi-level caching, Intelligent Cube architecture, and a fully optimized 64-bit technology, end-users can easily perform the most sophisticated analysis on data reaching through to the entire enterprise data warehouse while experiencing great performance and report interactivity. In addition, Intelligence Server's clustering, caching, monitoring, and control enable enterprises to scale to multiple terabytes of data and thousands of users as well as ensure a 24x7 fault tolerant environment.

6

MicroStrategy Web Architecture

MicroStrategy Web is the primary business user interface of the MicroStrategy BI platform. MicroStrategy's Web architecture provides a single, consistent interface to all users whether the BI application is departmental and internal, or an extranet application deployed to hundreds of thousands of users. MicroStrategy Web allows business users of any skill level to move fluidly between all styles of BI to satisfy their reporting, analysis, and monitoring needs.

Across all industries, styles of BI and deployment profiles, customers demand a BI Web architecture that is:

- High performing and scalable
- Secure, yet extremely simple to deploy and maintain
- As intuitive and interactive as the best desktop applications
- Easily integrated with existing systems, software and hardware

6.1 Core Design Principles

With over 15 years of experience supporting the most demanding BI applications, MicroStrategy delivers a unified BI platform and Web architecture that is proven to meet these customer requirements. MicroStrategy ensures customer success by rigorously adhering to five fundamental design principles:

- Be a part of a Service-Oriented Architecture for BI
- Use open standards in a layered architecture
- Build a platform-independent architecture on a single code base
- Ensure data security for all users
- Provide robust user interactivity with a no compromise, zero footprint interface

6.2 Optimize Performance with a Service-Oriented Architecture

MicroStrategy Web is the primary user interface of the broader MicroStrategy BI platform. Before discussing the Web architecture in detail, a brief review of how MicroStrategy Web fits into MicroStrategy's multi-tier BI architecture is useful. A Web-based deployment of MicroStrategy's BI platform consists of four primary tiers:

1. Data sources, which can be a relational database, ERP system, MDX cube, flat files or a combination of these.
2. MicroStrategy Intelligence Server, operating on a wide variety of hardware and operating system combinations.
3. MicroStrategy Web, operating on a wide variety of application server, Web server, and operating system combinations.
4. A Web browser operating on an end user's computer.

MicroStrategy's BI architecture optimizes the distribution of work across all four tiers in order to provide business users with the best possible performance. Jobs are distributed to the tier which can perform them most efficiently, minimizing the hardware and database resources required for the BI deployment.

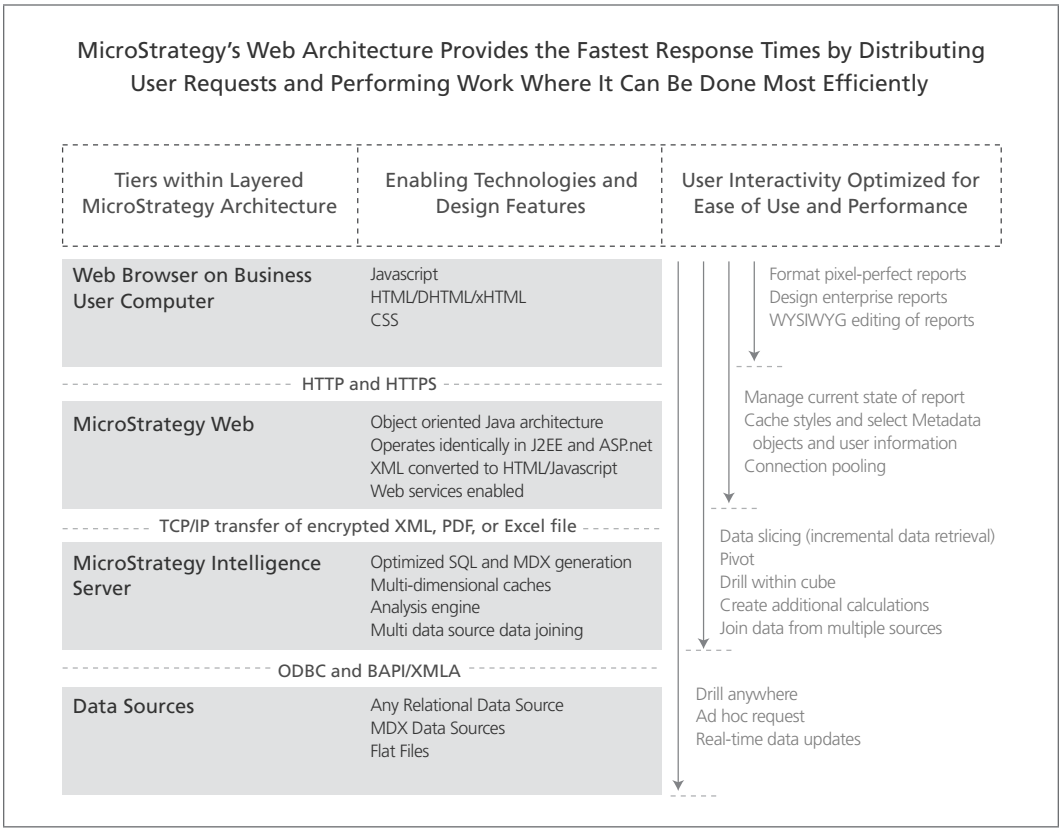


Figure 6.1 MicroStrategy's multi-tier BI architecture optimizes performance by distributing work to the tier which can process it most efficiently.

MicroStrategy Web's Role in the Multi-Tier Architecture

At the highest level, MicroStrategy Web serves two primary roles within the multi-tier MicroStrategy architecture:

- Translate end user actions into requests for information from Intelligence Server.
- Present end users with all the information, formatting, and interactivity provided by Intelligence Server.

The following high-level steps describe MicroStrategy Web's role in processing a user request:

1. Business users access MicroStrategy Web through their Web browser via a standard http(s) connection to a Web server. These user actions generate requests that are transmitted back to MicroStrategy Web.
2. MicroStrategy Web translates user actions into XML requests that are transmitted via TCP/IP to Intelligence Server. These requests may be for report results, metadata object definitions, or ad hoc calculations.
3. Intelligence Server retrieves data from data warehouses, operational systems, cubes, and the metadata repository, adds analytical richness to the data, and formats it.
4. Intelligence Server returns report results, formatting, security privileges, and other information to MicroStrategy Web in an XML format.
5. MicroStrategy Web translates the XML data into Web pages with all the appropriate information, formatting, layout, and BI functionality.

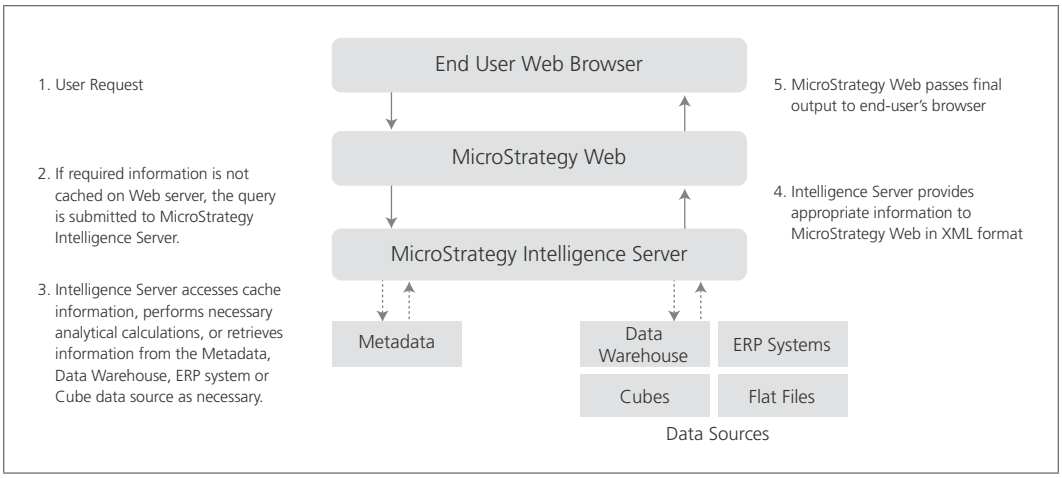


Figure 6.2 MicroStrategy Web handles all communications between end users and MicroStrategy Intelligence Server.

Many requests can be satisfied by MicroStrategy Web without communicating with Intelligence Server, and most user interactions and job requests occur with sub-second responses for hundreds or thousands of users simultaneously.

6.3 Open, Layered Design Optimizes Performance, Integration, and Customization

MicroStrategy Web's internal architecture consists of JavaBeans, Java classes, XML files, a servlet or ASP.NET container, and a thin layer of either ASP.NET pages or JSP pages. This architecture allows MicroStrategy Web to operate on virtually any operating system, application server, and Web server combination. MicroStrategy Web is installed on the server that hosts the application server.

The layered architecture separates the data access logic from the presentation and control logic, providing a more efficient internal workflow. Additionally, interface and application customizations are much simpler, since the cleanly layered architecture isolates modifications to a single layer of the architecture.

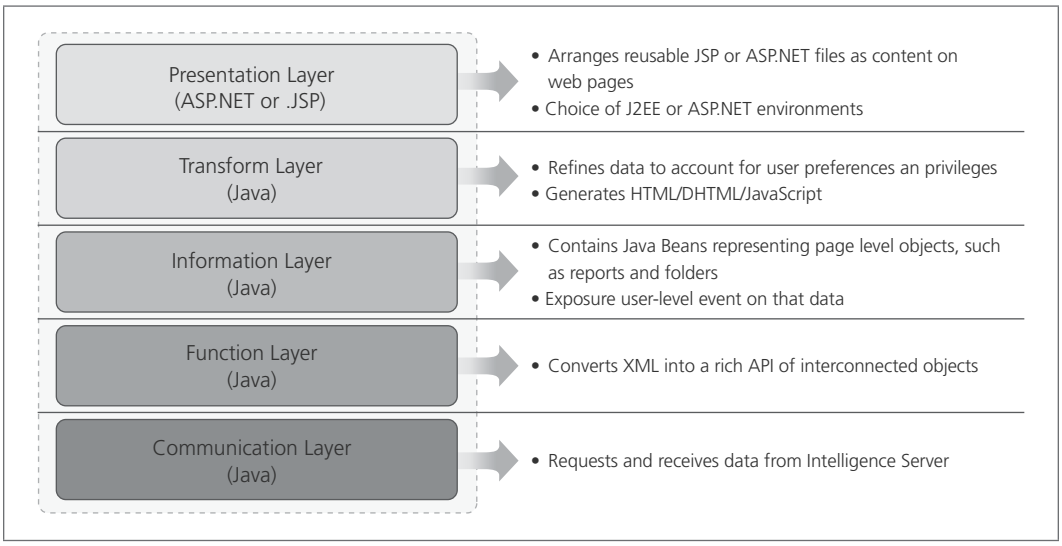


Figure 6.3 The MicroStrategy Web architecture provides distinct functional layers for more efficient processing of requests, and easy customization and integration.

The Communication Layer Interacts with Intelligence Server

The lowest layer of the Web architecture is the Communication Layer that manages the communication between MicroStrategy Intelligence Server and MicroStrategy Web. All messages are passed using an efficient XML structure that maximizes the throughput of Web requests.

The Function Layer Translates Requests into XML

The Function Layer sits above the Communication Layer and performs two primary jobs:

- Converts the XML returned by Intelligence Server into interconnected objects for use by other layers.
- Translates object interactions (for operations such as pivot, page-by, prompt, drill, and search) into requests for data from Intelligence Server submitted through the Communication Layer.

The Information Layer Instantiates All Web Data and Functionality

The Information Layer gathers the data and available functionality for a report, and forms self-contained MicroStrategy Java Beans, called "Web beans". A Web bean for a report provides the report data as well as the methods and properties necessary to pivot report grids, sort the data, and expose other report interactions. That is, in the context of a Model-View-Controller design paradigm, a Web bean is a miniature, combined Model and Controller object. It is a Model object in that it represents a high-level data concept (such as a report). It also fulfills the role of a Controller by coordinating the use of several Web objects to fulfill the collection of XML, and by managing the transform objects which are the View objects.

The Transform Layer Converts Web Bean Information into Output Formats

The Transform Layer combines BI information contained in the Web beans with transform styles to create report output formats. This layer also manages user preferences, and enforces user security privileges on the Web server.

The Presentation Layer Constructs the Final Web Page

The Presentation Layer defines the final Web page layout that is presented to the end user. It organizes the page sections such as header, page-by, grid, toolbar, footer, left panel and right panel, and identifies where the data from the Web bean should be rendered in those sections. A single Controller, used by either an ASP.NET or Java servlet, controls overall page execution, and manages all interactions between the presentation layer and the objects in the underlying layers.

Customizing the MicroStrategy Web Interface

Changes to the way MicroStrategy Web presents reports and BI functionality occur in external XML configuration files. No ASP.NET, JSP or Java coding is required for most customizations. The configuration files also provide a ready catalog of all changes made to the default configuration of MicroStrategy Web, resulting in customizations that are easy to maintain, extend, upgrade, or roll back to their default configuration. It is important to note that the configuration files do not need to be modified at all to fully deploy MicroStrategy Web.

MicroStrategy Web report results, functionality, formatting, and user interactivity are easily integrated in any commercial or homegrown portal and with other IT applications. MicroStrategy's layered Java architecture is compatible with industry standards such as:

- J2EE
- ASP.NET
- XML
- xHTML
- JSR 168
- WSRP
- SOAP

Style and Layout Parameter Files Determine Web Page Content

Modifications to Web page content are stored in the Style Catalog and Layout files using an xHTML format to embed MicroStrategy-specific tags into a familiar HTML structure. Only simple changes to file parameters are needed to perform common formatting and layout customizations such as rearranging or removing menus, toolbars, grids, and other GUI elements that compose the report page.

The Page Configuration File Catalogs All Web Pages

MicroStrategy Web contains Web pages specifically tailored to the activity performed on each page. Examples include the Login Page, Report Execution Page, Shared Reports Page, My Reports Page, Create Report Page and Create Document Page. The Page Configuration XML file catalogs all the pages available to the Web application in a format that is easy to understand and manage. Each JSP or ASP.NET file represents an independent, pluggable component that is not dependent on other JSP/ASP.NET files. The Page Configuration file externalizes the logic for combining these files into complete Web pages linked together for navigation.

Integration with Corporate Portals

MicroStrategy provides a Portal Integration Kit as a part of the MicroStrategy SDK. This kit contains deployment-ready portlets that require no additional programming when delivering MicroStrategy Web reports and functionality in third-party portal products. The Portal Integration Kit also contains sample code for a generic JSR-168 compatible portlet and sample code for deploying MicroStrategy within an Apache Struts framework.

Web Services Allow Loosely-Coupled Integration with Other Applications

The MicroStrategy SDK includes a Web Services Development Kit, which is composed of sample code and documentation describing MicroStrategy's Web services API's. The Web Services Development Kit is used as a starting point to deploy MicroStrategy reports as Web services for integration with other Web services applications. It utilizes the object-oriented layered MicroStrategy Web architecture and Apache SOAP Toolkit.

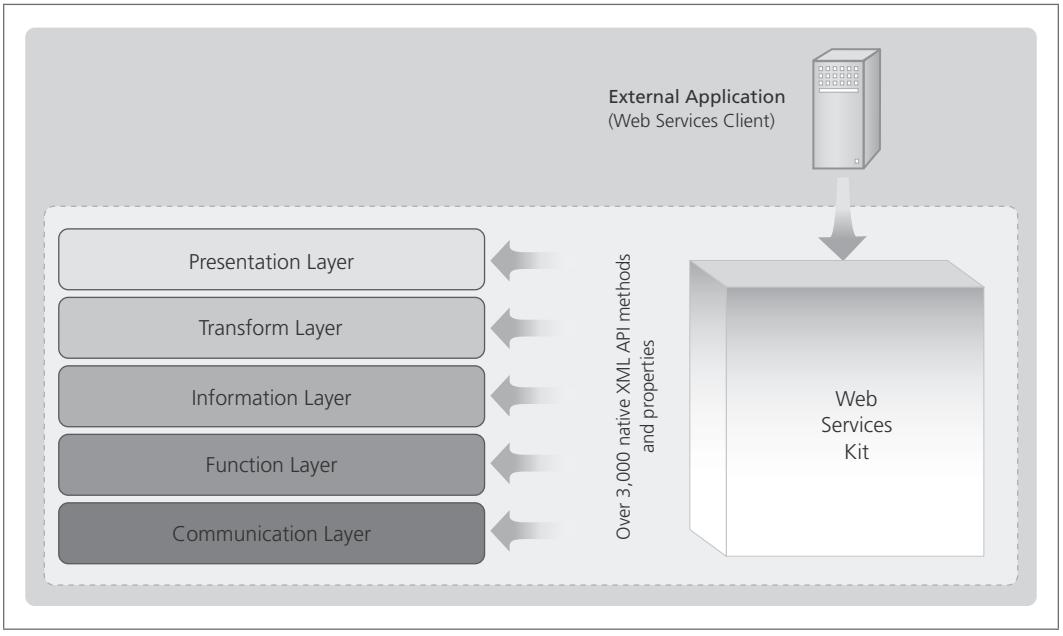


Figure 6.4 With the Web Services Development Kit, application developers use Web Services to integrate MicroStrategy BI applications with external applications or systems.

6.4 A Single, Platform-Independent Code Base Provides Flexibility and Consistency

MicroStrategy Web uses a single code base that is truly platform independent, and ensures that updates and upgrades occur on all supported environments simultaneously. The exact same reports, functionality, and user experience is delivered regardless of the choice of application server, operating system, hardware or ASP/JSP front-end. MicroStrategy Web supports all leading of application servers, Web servers, and operating systems.

Increases Flexibility and Reduces Future Risk

Many IT environments contain a variety of hardware and software, and require the flexibility to mix and match hardware and software across applications. Some organizations even choose to operate one hardware platform, operating system, and application server combination in their development and testing environments, and a different combination for their production systems. MicroStrategy's Web architecture enables this flexibility today.

Fast Response to Customer Issues and Enhancement Requests

Since 2003, MicroStrategy's Web architecture has consisted of a single unified code base across all supported operating systems, application servers, and Web servers. Other BI tools have different Web architectures for different styles of BI applications. These tools may have one Web interface and architecture for dashboards, and another interface and architecture for OLAP analysis. Compounding this problem, each interface is often compiled from separate code bases in order to run on different operating platforms. The same enhancements and issue fixes must be programmed many times into several different code bases. As a result, some fixes or enhancements appear in different interfaces or on different operating systems at different times, if at all.

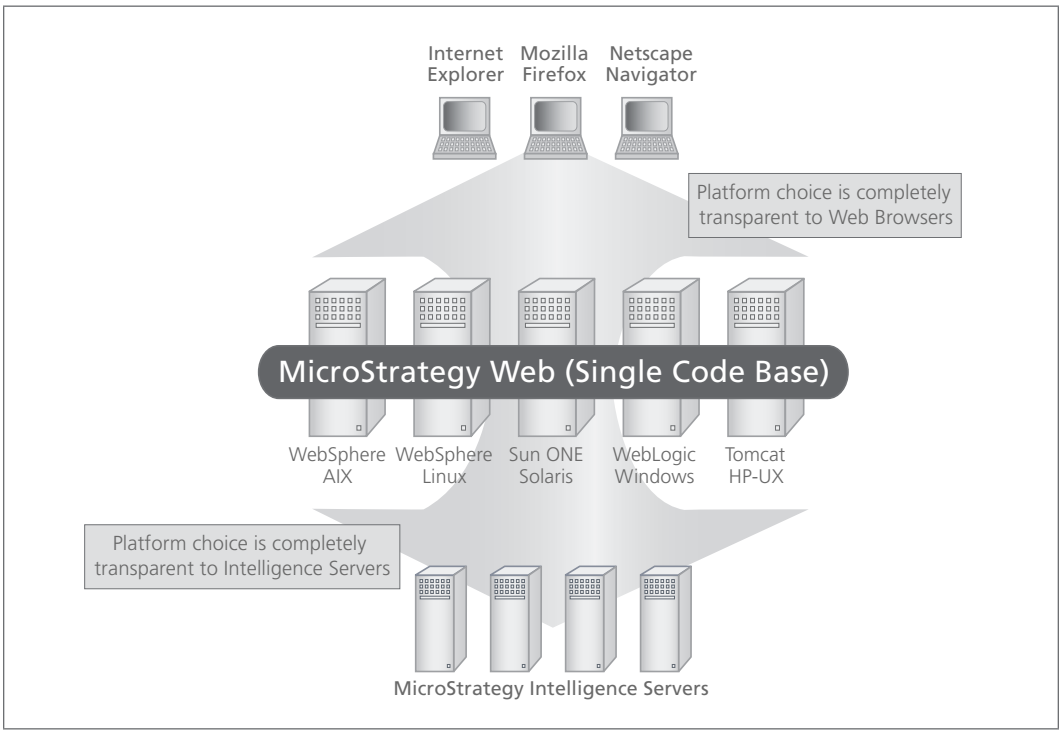


Figure 6.5 MicroStrategy Web's platform-independent, single code base provides maximum deployment flexibility.

6.5 All Data in the BI Application is Secure

MicroStrategy addresses security requirements with an elegant architectural design that is optimized for high performance and high scalability while providing information security. Five primary design principles ensure the integrity of the MicroStrategy Web architecture:

1. Secure communications across firewalls
   The MicroStrategy platform is normally installed on more than one server to distribute the BI workload. Secure communication across these servers is often governed by layers of firewalls constructed into Demilitarized Zones (DMZ). With three firewalls, two distinct DMZ's are created with one DMZ protecting the Web server and the second DMZ securing the infrastructure of the data source and Intelligence Server.

2. No database connection from the Web Server
   Web servers are the most common targets for hackers. It is vital that there is no direct access to the data warehouse from the Web server. MicroStrategy's BI architecture avoids this potential security problem by routing all communications with databases or other data sources through Intelligence Server. In this configuration, a hacker who gains access to the Web server does not also gain access to the data sources.

3. Single port control for data access
   Firewalls protect corporate information assets by limiting applications to certain computer network ports. The ports used by MicroStrategy Web and Intelligence Server for inter-server communication and for data access are fully configurable. As a result, intrusive port scans can be blocked by firewalls while letting users access critical business information.

4. No external Remote Procedure Calls (RPC) or Remote Method Invocation (RMI) calls
   RPC and RMI calls are hazardous because they allow hackers to access, and control remote and distributed computer processes. More importantly, these calls often allow anonymous access through separate, open ports in the firewall. MicroStrategy Web uses only XML to communicate with Intelligence Server eliminating the need for RPC or RMI calls completely.

5. No reliance on ActiveX, Java applets, or a specific Web Browser
   Many corporate firewalls prevent incoming ActiveX or Java applets as a standard security procedure. Unlike other BI products, MicroStrategy's BI platform does not make use of ActiveX, Java applets, downloaded compiled code, or a specific Web browser to deliver complete BI functionality to business users.

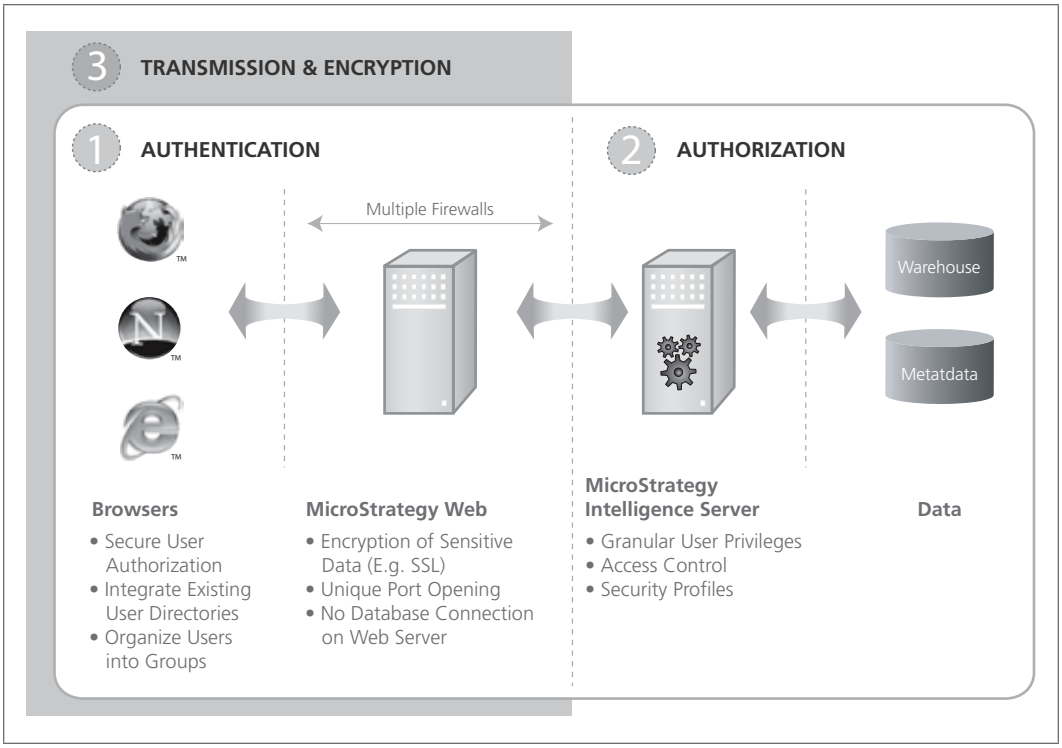


Figure 6.6 The MicroStrategy BI platform manages three layers of security: user authentication, user authorization, data transmission, and encryption.

6.6 Employ Leading Edge Technologies to Maximize Ease-of-Use and Interactivity

Zero Footprint Interface Maximizes User Adoption

MicroStrategy provides all BI functionality through a zero-footprint interface unlike other BI architectures that offer less functional zero-footprint interfaces alongside full-featured ActiveX or Java applet interfaces. Providing a full functionality, zero-footprint Web interface has been a guiding design principle for MicroStrategy Web since before 2000. MicroStrategy's zero-footprint Web architecture is well designed to satisfy common deployment requirements.

Requirements	Microstrategy Web's Features Support these Requirements
Access BI applications using any Web browser.	No dependence on a specific browser. Supports Internet Explorer, Mozilla Firefox, Netscape Navigator, and other browsers.
Deploy BI applications quickly and safely.	Users only need a URL address, which can be included on a corporate website or internal portal.
Restrict installation of additional software, applets, and plug-ins or ActiveX components on client machines.	No need to install any compiled code, or use plug-ins and ActiveX already embedded in Web browsers.
Minimize impact of Web browser security setting on the usability of the BI application.	User sessions are managed in cookies on the client or without cookies on the application server.
Conduct worry-free upgrades of BI Web server software.	MicroStrategy Web is only installed on the application server or Web server, and is updated to the latest software version within minutes.
Provide an interface compatible with the US Government's Section 508 accessibility requirements.	An HTML-only view of the interface is compatible with the US Government's Section 508, and designed to work with accessibility technologies such as screen readers.

Figure 6.7 MicroStrategy Web's zero footprint interface satisfies all user and administrative requirements.

MicroStrategy provides a single Web interface for all business reporting, analysis and monitoring applications. The same Web URL accesses all reports, dashboards, scorecards, analyses, and ad hoc queries. Navigation in the interface, running reports, and interacting with the results is identical for all styles of BI; users do not need to learn how to use different interfaces.

Groundbreaking Interactivity and Ease of Use

In 2002, MicroStrategy pioneered the use of XML, DHTML, JavaScript, and asynchronous communication in a zero-footprint Web interface that looked, and acted like a traditional desktop application. In 2005, the term "AJAX" (Asynchronous JavaScript and XML) was coined and popularized as a description for a category of applications that extensively use XML, DHTML, JavaScript, and asynchronous data loading and formatting for creating interactive Web applications. Although the term, "AJAX", is relatively new, its component technologies and characteristics evolved over multiple years into a general paradigm for Web development. For many years MicroStrategy has pioneered innovative uses of AJAX technologies such as DHTML and JavaScript in order to provide business users with the most intuitive and interactive BI experience on the Web.

Designed for the Business User with "Windows-on-the-Web" Simplicity

Business users of all skill levels find MicroStrategy Web intuitive and simple to use as they recognize familiar tools, techniques, and formats similar to any desktop application. With no prior training, MicroStrategy Web users easily access vital reports, perform insightful analysis, and monitor key business metrics using familiar techniques and Windows-like paradigms such as:

- Drag and drop actions
- Right click menus
- Drop down menus
- One-click functionality in toolbars
- Undo/redo buttons

Familiar Web Navigation Techniques

With MicroStrategy Web, all reports are organized into folders and subfolders with a navigation bar clearly indicating the location within the folder hierarchy. Reports and other objects can be found quickly using a familiar "Search" box. A single click of the mouse accomplishes common tasks like running a report, exporting a report, or drilling to additional detail. New reports can be saved in existing or new shared folders, or in private folders. Favorite reports can be bookmarked in a user's personalized "My Reports" area.

Floating Dialogs Instead of Intrusive Panels

Many BI tools provide input dialogs using panels that occupy valuable on-screen real-estate, and partially obscure the report. These panels cannot be moved, and restrict the business user's ability to interact with a report while still viewing the full context of the report. In contrast, for user input, MicroStrategy Web utilizes floating dialogs that are formatted to have a similar appearance to standard Windows dialogs. These dialogs can be moved anywhere on the screen, ensuring that business users never have to sacrifice important report context for interactivity.

Toolbars Provide Powerful One-Click Interactivity

The vast majority of BI interactions are available as toolbar buttons in the Web interface. The single-click actions are described by tooltips, which are displayed when the mouse pointer hovers over the toolbar button. The buttons are grouped into a series of toolbars, which can be hidden or shown with a simple right mouse click. Only users with the corresponding security privileges can use a given toolbar button.

Context-Sensitive Menus

A menu shows all available BI actions for a piece of data, an object, or an area of a report by simply right mouse-clicking on the data, object or area. MicroStrategy Web only displays the options that are permitted by the user's security profile, and that are appropriate in the current context. For example, when right-clicking on a report or folder name, common file management tasks such as copy, move, rename, or create a shortcut are available. When a business user right-clicks on a scorecard, dashboard or enterprise report, actions such as drilling, renaming, changing displays, copy, paste, and format are presented. This makes MicroStrategy Web as easy to use as an office productivity application.

6.7 Summary

For organizations planning to deploy their first BI application, and for those organizations consolidating tens or hundreds of applications into a common BI platform, MicroStrategy Web provides:

- An efficient, scalable, and high performance Web architecture.
- Independence from hardware, software, and operating environments.
- An extremely intuitive and easy-to-use interface.
- A single BI interface that supports all of an organization's users, BI applications, and data sources.

7

MicroStrategy Narrowcast Server Architecture

After business parameters and metrics are identified, and reports and documents have been created to depict the data in meaningful ways, users often need to run them regularly to monitor business progress. Often at this point, no interaction with the business intelligence system is necessary other than to retrieve the information in an efficient and timely manner.

7.1 Narrowcast Server is the Report Delivery Engine

MicroStrategy Narrowcast Server expands the MicroStrategy BI platform with automated content delivery capabilities. Enterprises benefit from flexible configuration options that support many different use cases such as:

- Batch reporting – sending out personalized operational reports, such as invoices, account statements, and business reports to analysts, partners, or customers.
- Business Monitoring – automated delivery of scorecards, dashboards, and threshold-driven alerts.
- Offline Analysis – delivering reports, including the underlying data, in business application formats such as macro-enabled Microsoft Excel workbooks or CSV data files.

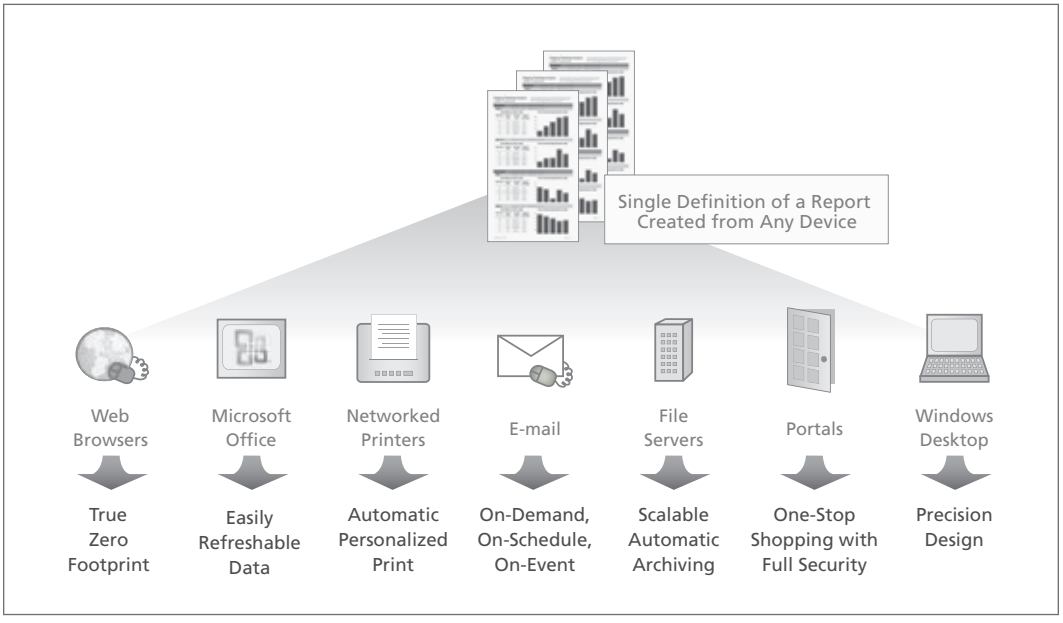


Figure 7.1 Narrowcast Server proactively delivers non-interactive information.

Information Delivery Design Tenets

Narrowcast Server is designed to efficiently and automatically deliver personalized information to any user in a timely manner. The underlying goals of the MicroStrategy Narrowcast Server are to:

- Scale from departmental solutions with tens to hundreds of users to support millions of users and messages per day.
- Support scheduled information delivery, exception-based information delivery, and ad hoc or on-demand initiation of content delivery.
- Allow a centralized rollout of information delivery or allow content and delivery process to be controlled by users themselves through self-subscription.
- Support the widest variety of delivery methods, with optimal formatting for each device and application.
- Support the broadest range of message content from small alert messages containing links for further analysis to consolidated reports from different information sources.
- Provide multi-locale support for international rollout.

These requirements are optimally addressed with a modular architecture and workflows that reflect the needs of the end user community. Once the system has been set up, it should run autonomously, and with minimal administration and maintenance.

Simplified Self-Subscription in MicroStrategy Web

Narrowcast Server contains default services and schedules that enable automated report delivery directly from the MicroStrategy Web interface. A user would run a report, provide answers to any prompts, and perform any additional analysis. After the final analysis is saved, it is available for immediate or scheduled delivery. The delivery schedule, format, method, and address, along with a subject and message, can be specified with each subscription. A subscription is automatically created in the metadata repository containing this information.

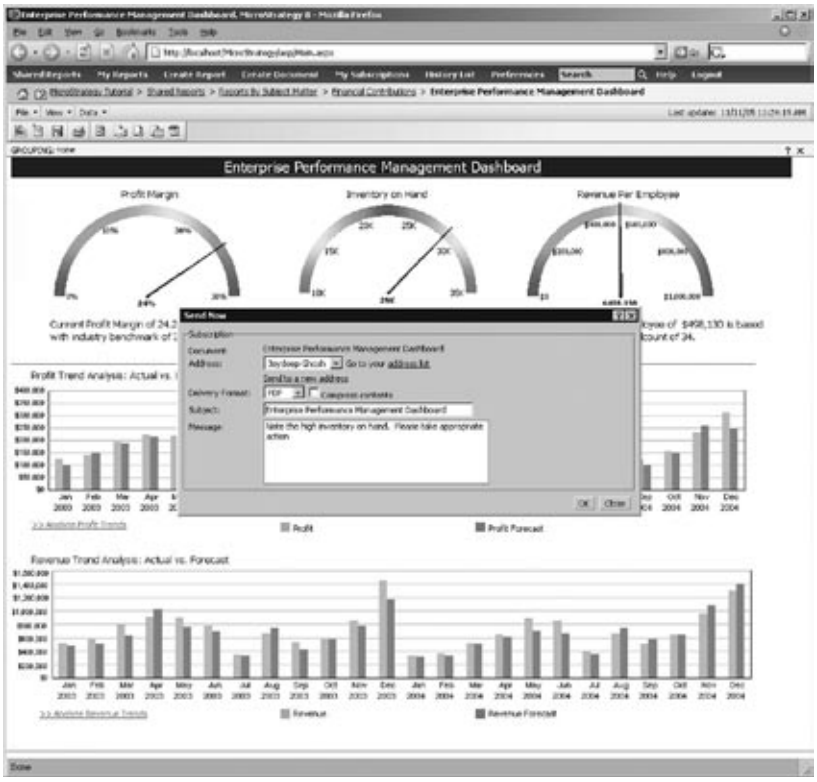


Figure 7.2 Users can subscribe to receive or send reports to others through MicroStrategy Web.

Narrowcast Server constantly evaluates all schedule patterns, and calculates a list of execution times for the appropriate services. At a specified time, Narrowcast Server starts the execution by retrieving all subscriptions for the schedule from the metadata. After evaluating all subscriptions for the associated schedule, it retrieves the information in the format that was specified in the subscription from Intelligence Server. Then it completes the message, and delivers it to the specified address. Delivery is possible via email, to printers, and to file servers.

7.2 Architecture and Workflow

Enterprises deploy report delivery applications in a top-down or bottom-up fashion. The top-down method is a managed process in which information is delivered without immediate control over the process or information for the recipient. Examples are the email delivery of invoices and account statements to customers, or sales performance overviews and customer information to account representatives.

The bottom-up approach, referred to as self-subscription, gives control and personalization capabilities to the recipient. For example, individuals define the scheduled delivery of a weekly business report to a local printer for a regular meeting, or the alert-triggered email delivery of a report that monitors a business process.

Both deployment methods serve application-specific needs and therefore are well supported. The following capabilities enable top-down deployments:

- Manually subscribing recipients as part of the deployment process
- Importing subscription information in bulk
- Dynamically creating subscription lists in an external application at runtime
- Users subscribing other users to reports and documents in the MicroStrategy Web interface

Bottom-up deployments are possible using:

- The MicroStrategy Web interface, which allows self-subscription for email, printer, and file deliveries
- The Subscription Portal, which enables self-subscription to delivery services with optional personalization for end users

Many MicroStrategy report delivery applications use both methods to deploy information to different user communities.

Functionality Broken Down Into Components

Narrowcast Server comprises four main components – a metadata repository, a delivery engine, an administration console, and a subscription portal.

- The Metadata Repository is a central data store that maintains all information about delivery services and subscriptions.

- The Delivery Engine is responsible for creating, formatting, transmitting, and administering messages. It is comprised of the following components:

    - Distribution Manager – determines when a service is executed, assigns an execution engine, queues service requests, and provides failover support
    - Execution Engine – determines the service definition and subscriber information, and directs the service execution
    - Information Source Module – retrieves information from various data sources

- Formatting Module – formats the data for delivery to the requested device
- Information Transmission Module – transmits the formatted documents using the appropriate protocol for the requested device
- Logger – logs execution statistics and debug information for every step in the service execution process

- The Narrowcast Administrator provides a unified console to monitor and manage the delivery engine remotely. It also provides a development environment for the creation, testing, and deployment of services using its built-in object management functionality.

- The Subscription Portal is responsible for Web-based delivery of personalized messages. It also provides Web-based subscription capabilities for delivery services, including ways to capture preferences for message personalization, and is also responsible for gathering user profiles. The Subscription Portal comprises the following components:

  - Subscription Portal – accesses report information and subscribes to services in a Web interface
  - Subscription Site Designer – defines the look and feel of a Web-based subscription portal
  - DataDirect SequeLink® ODBC Socket server – provides managed access to the metadata repository

Each component of Narrowcast Server plays a role in creating, formatting and delivering services to subscribers.

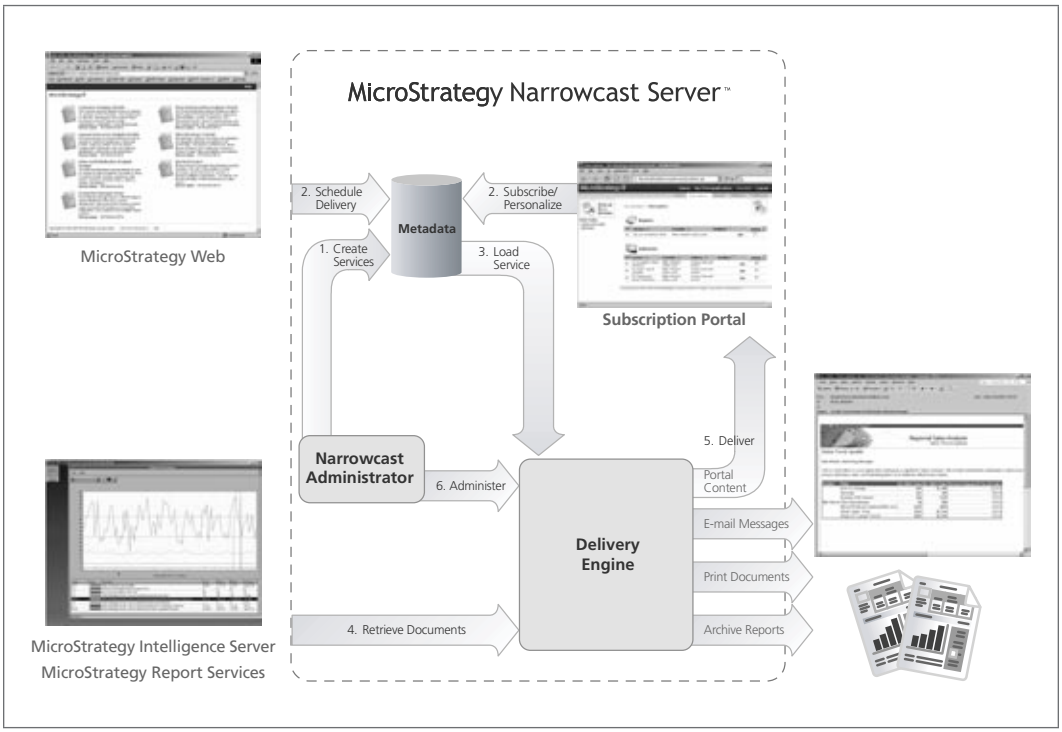


Figure 7.3 Narrowcast Server's four components provide flexible deployment options.

Workflow of Data through Narrowcast Server

A delivery service is used to control and map the flow of data from its source to the recipient.

At a high level, Narrowcast Server receives reports and documents in HTML, Microsoft Excel or Adobe® PDF format from Intelligence Server and delivers them to email recipients, Web portals, file servers, and printers. The following workflow explains the architecture, as depicted above, when executing a delivery service through Narrowcast Server.

1. Narrowcast Server provides delivery services facilitating delivery of reports and documents scheduled in MicroStrategy Web. Additional delivery services can be created by developers using the Narrowcast Administrator console. These services, including sub-components and schedules, are maintained in the metadata.
2. If Web-based self-subscription to delivery services is enabled, MicroStrategy Web and/or the Subscription Portal provide self-subscription capabilities. Users can set up and maintain subscriptions to personalized reports and documents or subscribe to receive messages from delivery services. These subscriptions are stored in the metadata.
3. When the service execution is triggered, the Delivery Engine first loads the service definition from the metadata. It also retrieves all the subscriptions for the service, which can exist statically in the metadata, or which can be retrieved from Intelligence Server or another information source as a Dynamic Subscription Set.
4. The Delivery Engine will then request personalized information from Intelligence Server by executing reports and/or documents. This information will be formatted in the way that was specified by the service designer or a self-subscribing user. This content will be used to create the final messages.
5. The finalized messages are delivered to the Subscription Portal, printers, file servers, email server, or other output device for delivery via email.
6. The Narrowcast Server Administrator allows full administration of the Delivery Engine. Service executions can be triggered, monitored and canceled, and machines can be dynamically added or removed depending on capacity requirements. Administrators can receive operational status messages and alerts via email.

Configuration Options Ensure Secure, Reliable Report Delivery

The primary reason for separating Narrowcast Server into four components is to give enterprises flexibility in deploying the components on different machines. While the simplest configuration is to install all Narrowcast Server components on a single machine, there are good reasons to opt for configurations that deploy the components on multiple machines or nodes in an enterprise environment.

- As Narrowcast Server provides a non-interactive reporting environment, it often requires less stringent response time requirements than the interactive Web environment. Narrowcast Server also takes over the formatting tasks of Report Services documents when it is installed on a separate machine, reducing the load on the interactive environment.

- Many enterprises have strict rules protecting sensitive data, especially when applications are deployed to extranet users. Narrowcast Server is designed to support multi-tiered configurations with application and Web components located on separate networks divided by firewalls.

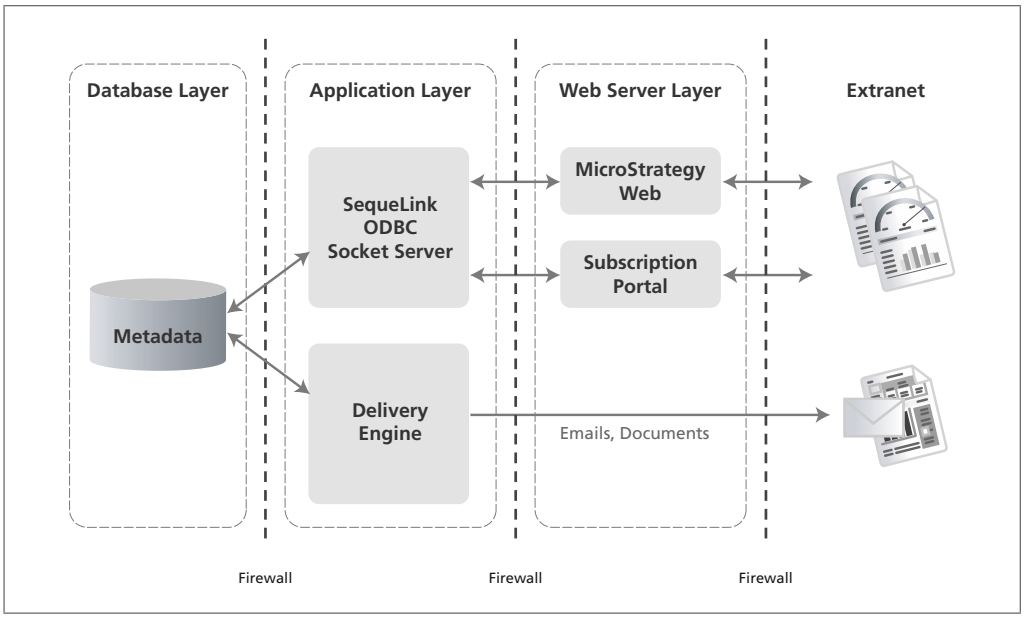


Figure 7.4 The 4-tiered configuration of Narrowcast Server components in a secure extranet deployment.

- When Narrowcast Server processes large volumes of subscriptions, it divides the work into smaller sets called segments. These segments are processed in parallel, and significantly enhance message throughput. Narrowcast Server implements a divide and conquer algorithm that distributes these segments effectively between the machines in a cluster, and also allows granular tuning of the delivery process. For this to occur, the Delivery Engine is broken down further into Distribution Manager and Execution Engine functionality. One node in a Narrowcast Server cluster will take on the role of Distribution Manager. This node directs the delivery process by breaking down the work of executing a service into segments, and managing their processing on the other nodes that take on the role of Execution Engine.

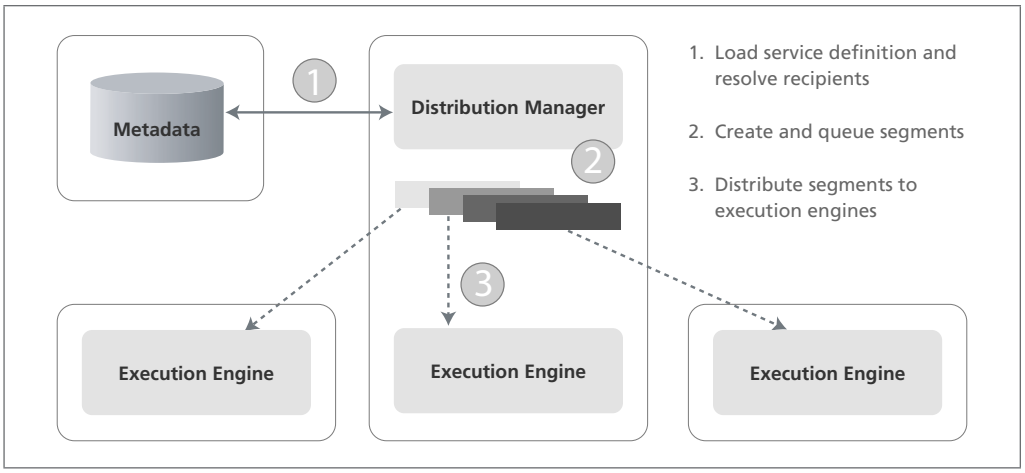


Figure 7.5 Segmentation allows efficient load balancing in a Narrowcast Server cluster.

Narrowcast Server contains full clustering and failover capabilities, ensuring reliable delivery of information. Execution Engines back each other up using a round robin algorithm. An additional node can be designated as the backup Distribution Manager for complete failover support.

7.3 Creating Report Delivery Services

Delivery services are created in the Narrowcast Administrator console. All service definitions are stored in the metadata repository. A service is defined as a combination of a message definition, recipient information, delivery information, and a schedule.

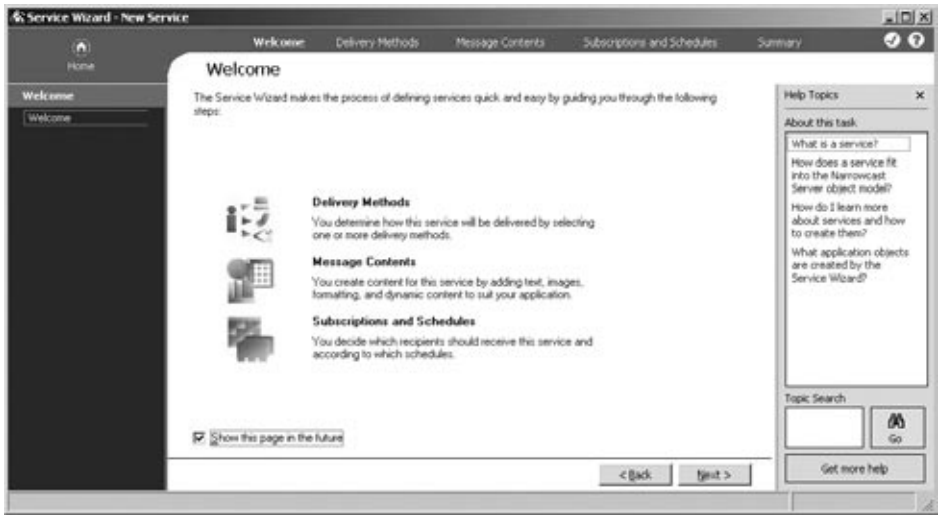


Figure 7.6 The Narrowcast Administrator provides an intuitive Service Wizard for creating services.

The Service Wizard guides the service designer through a sequence of three simple steps to successfully create a service:

1. Select the delivery method:
   - [a] Email, Web portal, printer, file server

2. Define personalized message content.
   - i. Define independent layouts for different message parts.
     - Email subject, email body, and attachments
   - ii. Specify the appropriate document format for optimal delivery.
     - HTML, Microsoft Excel, PDF, plain text, XML, and compressed zip file
   - iii. Specify personalized dynamic content using MicroStrategy reports and documents.

3. Subscribe users to delivery schedules:
   - i. Choose the service recipients' delivery addresses
     - Static subscriptions from the Subscription Book or dynamic subscription rules to determine recipients at runtime
   - ii. Select the schedule that triggers the report delivery
     - Time intervals, external events, and threshold alerts

Flexible Delivery Services Handle Large Report Volumes

Narrowcast Server's delivery services take advantage of efficiencies that exist when similar information is sent to a large number of recipients. The features listed below provide the industrial strength needed for enterprise deployments:

- Specify processing and delivery of messages in any supported language and locale. A single service can make content available in many languages, such as English, German, and Korean.
- Format information specifically for the target delivery devices. For example, emails sent to mobile devices with

small screens should be designed differently from emails sent to desktop users with large screens.

- Combine content from several information sources in the same message. Information can come from different MicroStrategy projects or, by using custom modules, from other IT systems.
- Optimize the information gathering process by taking advantage of report caches and Personalized Page Execution mode (PPE), also called report bursting. In this mode, result pages of one large Intelligence Server report are distributed to different subscribers according to the page-by settings. This approach significantly reduces the number of jobs submitted to Intelligence Server, making it possible to scale report delivery operations.
- Distribute the formatting process for Report Services documents to the Narrowcast Server machines. This results in an optimally balanced system, improving response times for interactive Web users and for Narrowcast Server deliveries.
- Import recipients from third-party systems dynamically using Dynamic Subscription Sets.
- Custom build Information Transmission Modules which send documents and reports to fax machines, FTP servers, or any other output devices.

Smart Alerting using Dynamic Subscription Sets

Report delivery is taken to a new level via Delivery Services that automatically determine who should receive critical information. Combining the analytical capabilities of Intelligence Server with Narrowcast Server's ability to reliably distribute information creates an Information Radar – a BI system that can check thousands of business parameters concurrently, and alert stakeholders only when necessary.

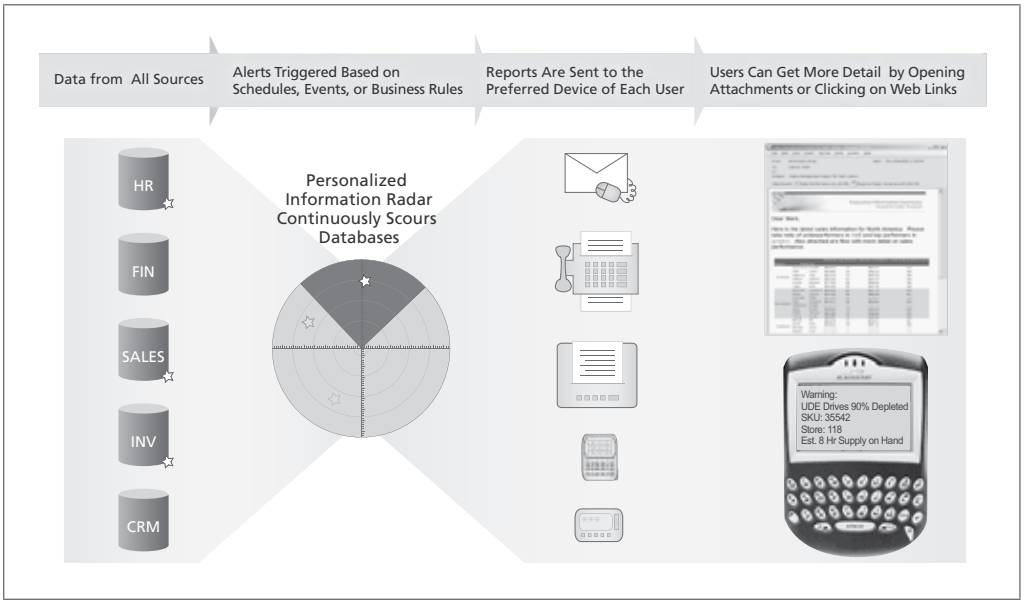


Figure 7.7 MicroStrategy's "Information Radar" lets organizations stay on top of business parameters.

Efficient alert-driven deliveries are implemented using Dynamic Subscription Sets. Instead of applying a simple list of recipients to a service, Dynamic Subscription Sets generate a set of recipients according to the information retrieved by a report or limited by a filter. The evaluation criteria can be as complex as the analytical engine allows. Individual personalization of parameters is possible. Dynamic Subscription Sets provide the following benefits.

- Maintenance of subscription lists is reduced. For example, a weekly delivery service sends account details to the sales team. The content that each team member receives changes over time – new hires are added, members leave the team, and accounts change hands. A Dynamic Subscription Set picks up these subscription changes directly from the data warehouse or sales force automation system.

- Only relevant information is sent to recipients, eliminating unnecessary deliveries. Dynamic Subscription Sets are generated with the analytical capabilities of the MicroStrategy BI platform. The filtering criteria used in the report determine the list of subscribers. For example, an inventory shortage alert is delivered only to certain store managers for products with low stock levels or outages. The list of store managers who receive this regular alert is created every time the service is run.

Extensibility Using Modules

Narrowcast Server is an open and extensible information delivery platform. The foundation of this extensibility is the fully documented Narrowcast Server API and the concept of modules. Modules extend the delivery platform in three areas:

- Information Source Modules give access to non-MicroStrategy data sources, such as SAP BW and other ERP systems, Web pages from Internet providers and portals, and non-relational content such as flat files and images.
- Document Formatting Modules add new formatting capabilities to Narrowcast Server messages. For example, reports can be formatted into Microsoft Word or PowerPoint documents.
- Information Transmission Modules deliver information using new communication protocols. Examples include document delivery to Fax devices or file transfer protocol (ftp).

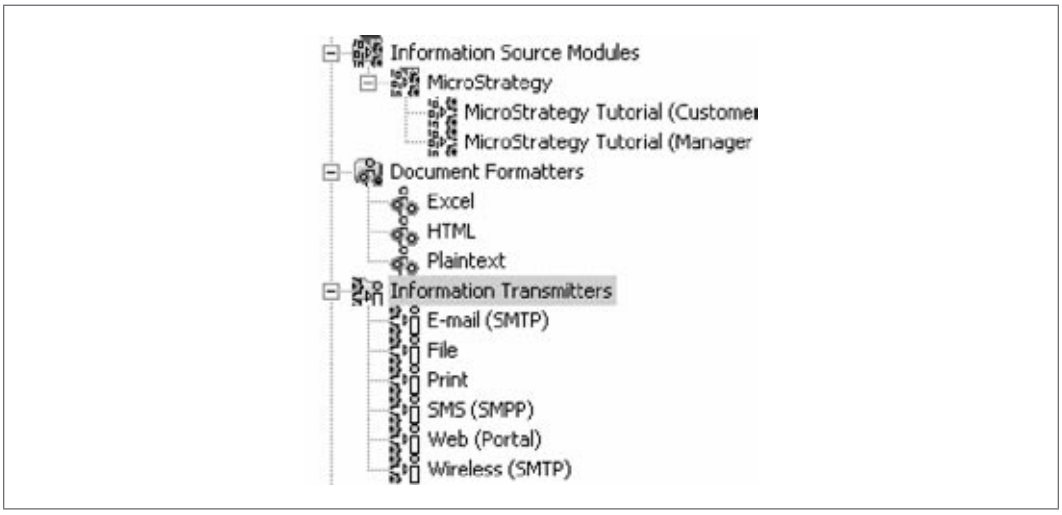


Figure 7.8 Modular design allows extending Narrowcast Server functionality.

The MicroStrategy SDK includes sample code and documentation to facilitate rapid development of custom Narrowcast Server modules. Once included in Narrowcast Server, these custom modules become available for use in all report delivery services.

7.4 Deploying Non-Interactive Content using the Subscription Portal

The primary function of the Subscription Portal is to deploy non-interactive content and documents defined in delivery services over the Web. The Subscription Portal is configured and maintained using a wizard called the Subscription Site Designer. The Subscription Site Designer itemizes which delivery services are published on the portal, and which personalization options are available for each service. The appearance of the Subscription Portal interface can be changed to conform to corporate standards. SSL encryption ensures a safe transmission of corporate data to remotely located recipients.

The Subscription Portal offers self-subscription capabilities for report delivery services, and enables users to manage their profiles and preferences. Users complete four steps when subscribing to delivery services in the Subscription Portal:

1. Select the report delivery service they wish to receive
2. Select the device on which they want to receive the message
3. Select the schedule or trigger for service delivery
4. Select personalization settings that should be applied to the data

The self-subscription paradigm used with the Subscription Portal or MicroStrategy Web enables deployment to large, diverse user communities. Because the subscribers create their own profiles and preferences themselves, administrative overhead is reduced significantly.

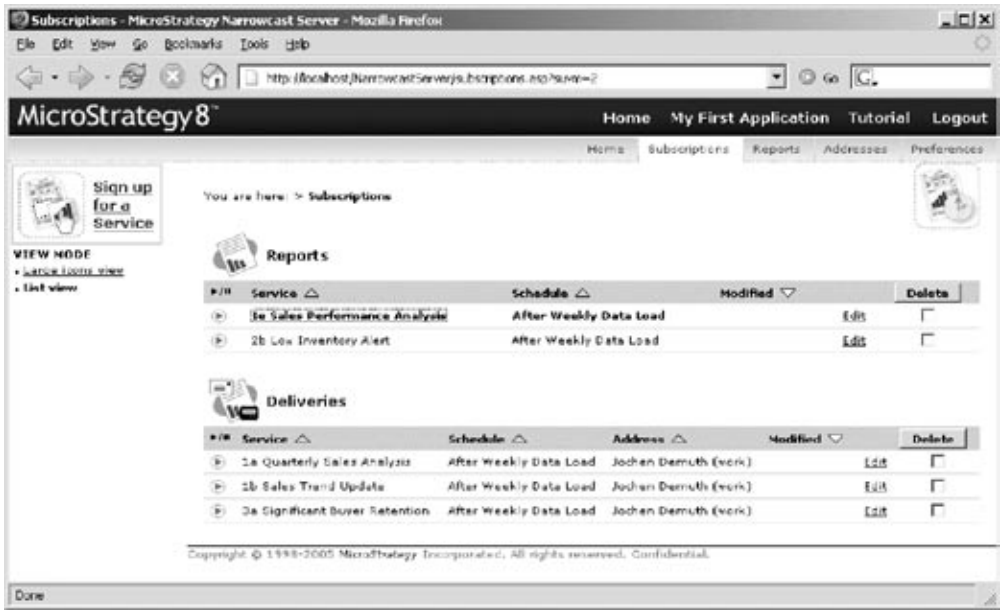


Figure 7.9 The Subscription Portal allows users to personalize and subscribe to services from any Web browser.

7.5 Architected For Enterprise-class Report Delivery

MicroStrategy Narrowcast Server provides a flexible architecture that is designed to support all batch reporting, monitoring, and offline analysis applications. Its range of features enables a robust, scalable, and efficient rollout to BI users, partners and customers alike. The integration into MicroStrategy Web empowers existing BI users to use MicroStrategy BI more efficiently. The flexible concept of delivery services supports both top-down and bottom-up subscription management. In sum, Narrowcast Server makes it possible to deliver personalized information to any user at any time, empowering users to make critical business decisions in a timely manner.

PLATFORM FUNCTIONALITY

8 Security

Most organizations describe their BI projects as strategic or mission critical. As they make their systems available to wider internal and external audiences, information must be safeguarded from potential security threats such as unauthorized access, data tampering, and eavesdropping. A truly secure BI system must have a multi-layered security paradigm; security in the business intelligence platform alone is insufficient since there are multiple points of access to the data warehouse. A wide range of security products and techniques are available for protecting the overall implementation, including:

- Intrusion detection software
- Antivirus software
- Firewalls and proxy servers
- Authentication schemas
- Data warehouse connections
- Security views in the data warehouse
- Application-specific security
- Control over application functionality
- Cell-level security for reports
- Encrypted data storage and transmission

The MicroStrategy BI platform integrates seamlessly with these security measures, and includes its own security features.

8.1 Securing Business Intelligence Applications

Securing BI applications and the sensitive data they contain requires resolving three fundamental questions:

- Is the person trying to access the BI system a known user, and can his or her identity be verified?
- For what functionality, reports, metrics, and data is the user authorized?
- How can the system respond to user requests in a manner that prevents the data from being accessed by anyone other than the authenticated and authorized recipient?

More generally, these questions may be abstracted into three critical areas of enterprise software security:

1. User Authentication
2. User Authorization, including three key subcomponents:
    i. Application functionality security
    ii. Access control lists
    iii. Data security
3. Internet architecture and transmission security

MicroStrategy provides a BI architecture that enables corporations to confidently address all security requirements in a manner that maximizes flexibility and scalability, and minimizes administrative effort.

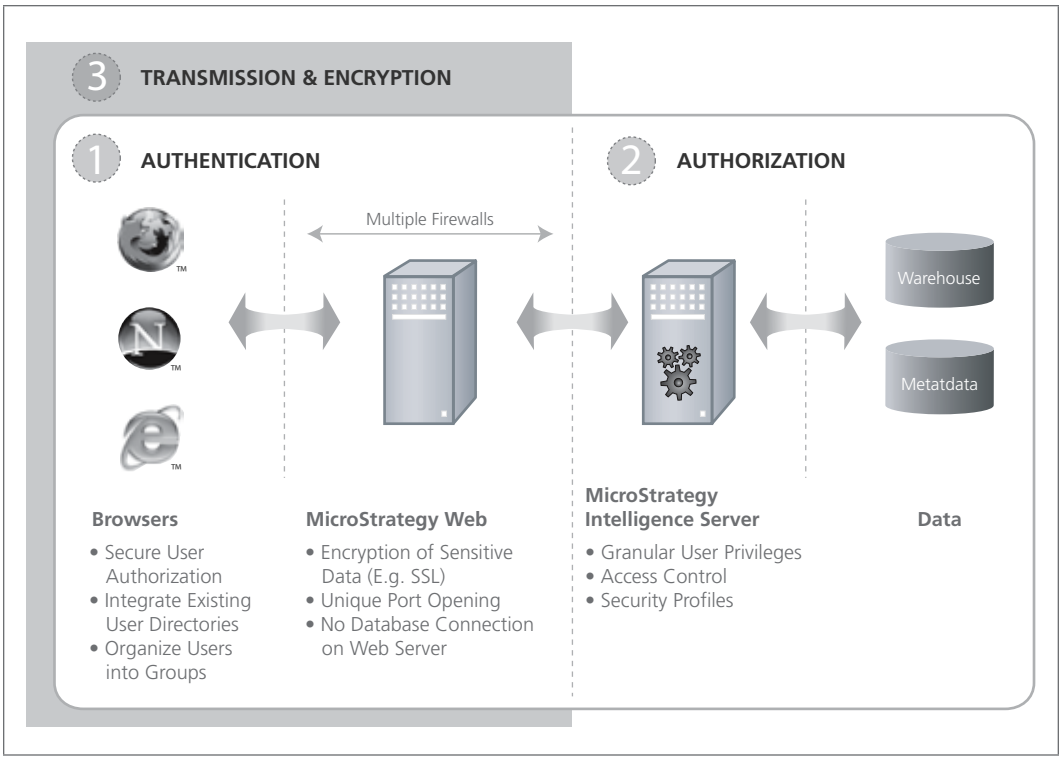


Figure 8.1 Security must be deep and pervasive throughout the architecture.

8.2 User Authentication

User authentication is the process of confirming a user's identity. The MicroStrategy BI platform maintains a profile for each user of the BI system. These profiles can be created using a graphical interface, scripts with textual commands, or synchronized directly with Windows or LDAP user directories.

Validation of user credentials during initial access can be performed:

- Within MicroStrategy using "Standard" Authentication
- Outside of MicroStrategy using "Linked" Authentication with a third-party such as Windows NTLM, Active Directory, LDAP, a Relational Database, or SAP
- Through customization via an External Security Plug-in

Standard Authentication

With standard authentication, all user logins, passwords, and security settings are stored within MicroStrategy metadata in a hashed format. Standard authentication is manual process where a user is prompted for his or her user login and password each time a new session is started. The login and password are sent to MicroStrategy using a 160-bit one-way encryption algorithm, and is compared to the login and password stored in the MicroStrategy metadata repository; and if a match is found, the user is granted access to the system. Throughout the user's session, all security settings associated with the user login will remain in effect, and be applied by MicroStrategy in a manner that is fully transparent to the user.

Linked Authentication

Many corporations have a single, centralized security directory which governs user access to internal systems and applications across the entire enterprise. MicroStrategy integrates seamlessly with security directories through linked authentication, which is also referred to as external authentication. Benefits of linked authentication include reduced administration and maintenance of users in the BI system, and increased ease of use for end users. Linked authentication is often automatic, meaning the MicroStrategy system detects the user credentials, and automatically validates those credentials against one of the following security systems:

- Lightweight Directory Access Protocol (LDAP)
- Microsoft Windows Active Directory®
- Windows NT® LAN Manager (NTLM)
- Database security
- SAP BW

With automatic, linked authentication, users typically use a single login to access both the network and the MicroStrategy BI platform. Business users are able to move freely between all BI applications and other enterprise applications without maintaining multiple logins and passwords. This single sign-on capability further simplifies administration and maintenance of security profiles and groups, and complementary authentication technologies such as digital certificates are applied transparently to the MicroStrategy BI platform.

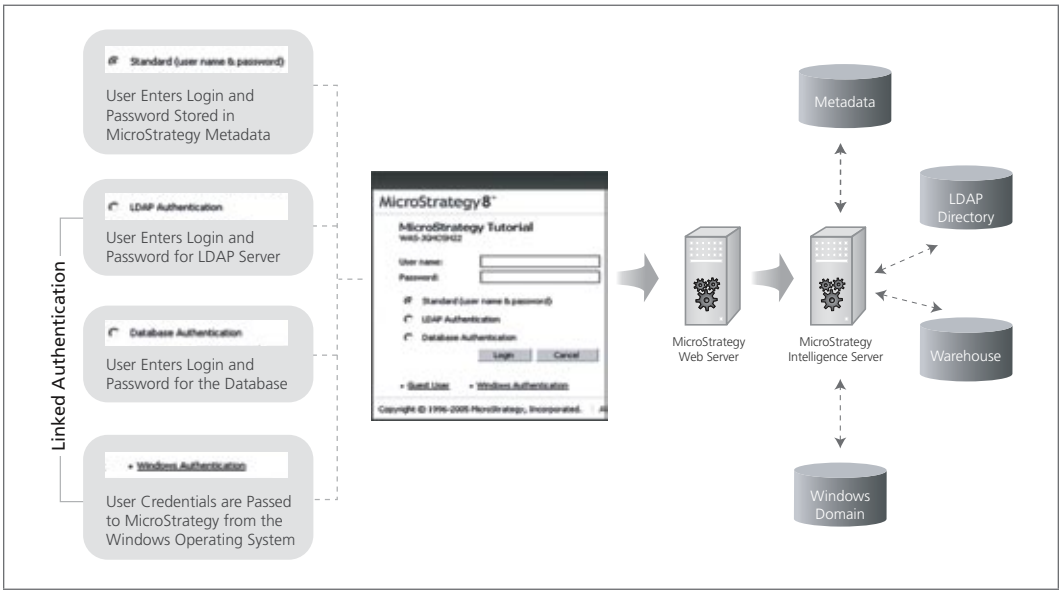


Figure 8.2 User authentication is based upon user security profiles in MicroStrategy's metadata or links to other security systems such as the database, LDAP or Windows.

Integrating with Single Sign-On Solutions

MicroStrategy also provides methods for integrating with customized security authentication schemes already in use within an organization. Corporate Portals and identity management products like eTrust™ Identity and Access Management (formerly Netegrity) or Oracle Identity Management (formerly Oblix) are common systems with which to implement single sign-on integration.

8.3 User Authorization

Within any enterprise application, it is crucial to distinguish between users based on each individual's knowledge, business needs and security level. Users are not granted the same rights to application functionality, reports, and data. Once a user has been authenticated, the BI system still must determine and enforce security policies governing the functionality, reports, and data for which the user is authorized. Authorization refers to the three-dimensional process by which an application determines:

- Application functionality privileges
- Object access permissions
- Data access security

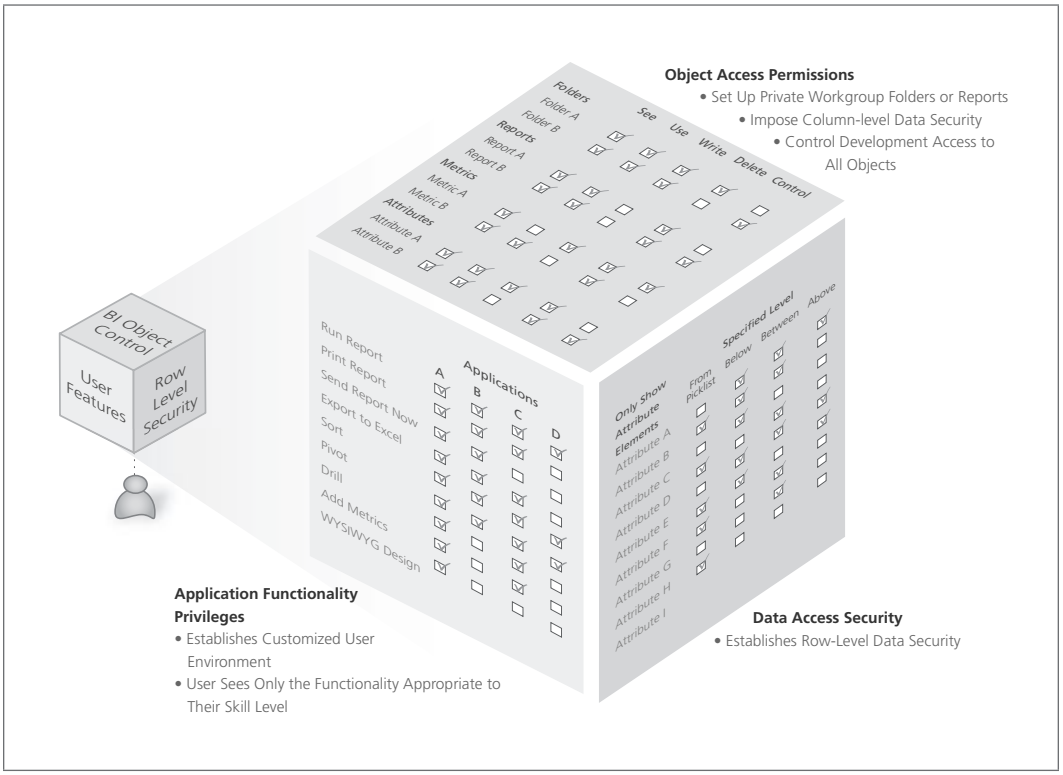


Figure 8.3 MicroStrategy's comprehensive application, object access, and data access security provides granular control for BI administrators.

Application Functionality Privileges

Users fall into various types, including casual users, power users, application developers, and administrators. Depending on their levels of sophistication, certain users might need basic functionality such as running reports and sorting the results, while power analysts might need to create their own analyses and publish them. Application developers need object creation privileges, and administrators require specific monitoring and management functionality. MicroStrategy employs over 130 privileges to assign application functionality to user groups, user roles, and individual users.

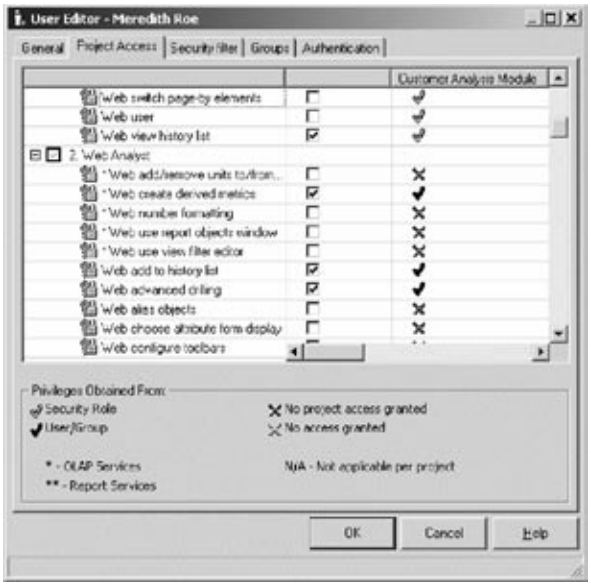


Figure 8.4 Desktop has a graphical interface to assign privileges to users.

These privileges can be assigned to users, user roles, and user groups through a graphical interface, and through text-based commands that can be run in batches with Command Manager.

Such fine-grained control ensures that all users access the MicroStrategy platform according to their level of proficiency. With administrators setting various levels of application functionality security, users can start using the BI applications with minimal training. Over time, administrators may grant users more privileges as they become more experienced.

Object Access Permissions

Individual MicroStrategy metadata objects are governed by their own security permissions, called Access Control Lists (ACL). Each data abstraction object, business abstraction object, report component, and report definition may have its own unique ACL, which grants users or user groups a set of privileges for the object. To simplify application maintenance an ACL can apply to many objects. MicroStrategy's object-oriented metadata allows ACLs to be inherited by child objects and applied recursively. Seven permissions can be combined to grant or deny object behavior to user groups or to individual users.

- Browse – view the object in a folder and viewer
- Read – view the object's properties (definition and, ACL settings)
- Write – modify the object's definition but not the ACL settings
- Delete – delete the object from the metadata repository
- Control – modify the object's ACL settings and take ownership of the object
- Use – reference the object when creating or modifying other objects
- Execute – reference the object when running documents and reports

These permissions have been arranged into predefined groups that reflect the most commonly used sets of permissions as follows:

- View – contains Browse, Read, Use, and Execute
- Modify – contains Browse, Read, Write, Delete, Use, and Execute
- Full Control – contains Browse, Read, Write, Delete, Control, Use, and Execute
- Denied All – explicitly denies all object access

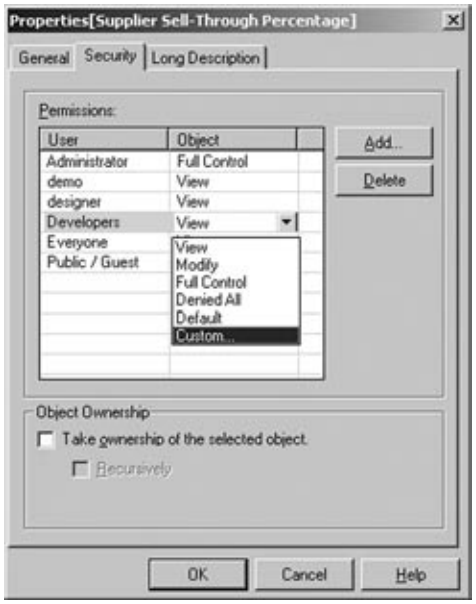


Figure 8.5 Users and user groups are granted or denied access privileges through object properties.

For example, when a report contains many metrics, each metric's "Execute" permission determines whether a user can view that metric in the report. Developers can minimize the number of reports they create and store in the metadata repository by making use of this feature. When two users run the same report, they will receive different versions of that report from a single report definition in the metadata, based on their ACL settings.

Different users may even be given access to completely different subsets of the data and business abstraction objects. For example, corporate expense attributes, metrics, and reports can be restricted to managers and finance department users. ACL settings can also be applied when drilling so that some users can drill down to detailed information while others cannot.

Securing Data using Database Security

In a database, security restrictions for database logins can be placed on tables, rows, and columns. MicroStrategy's BI platform accesses data sources using database connections. Separate database connections can be created to access the same data source with different logins. MicroStrategy users and user groups are linked to database connections using connection maps.

All users allocated to a database connection will log in to the database with the same credentials, and will be subject to the security setting in the data source. Furthermore, database views may include a restriction by database login in their definition. This login, obtained from the database connection information, limits the rows that are selected by the view when processing queries. These security views provide row-level security for every query submitted by the user. Since an administrator defines this security view inside the data source, all query tools accessing the data source with a particular login will use the view. The SQL statement used to create the database view can also be used within MicroStrategy to define a logical table in the metadata.

The main disadvantage of using database views to manage security is that performance may degrade because the view is processed at run-time. Security views must also be recreated every time new data is added to the tables in order to optimize the query that defines the view.

Securing Data with Security Filters

Security filters provide a method for ensuring cell-level data security. All the filtering sophistication available in MicroStrategy can be used to limit the data that a user or user group can access. For every data source request, including documents, reports, and prompt lists, additional filtering criteria is automatically added to the query to restrict the result set to information that the user is permitted to access.

Take an example of a new Northeast Regional manager who was transferred from the South at the beginning of 2005. She needs access to all Northeast data to-date, but may also need access to South data for previous years. As a result of this security filter, all data requests will be restricted to the Northeast region, and also the South region for prior years.

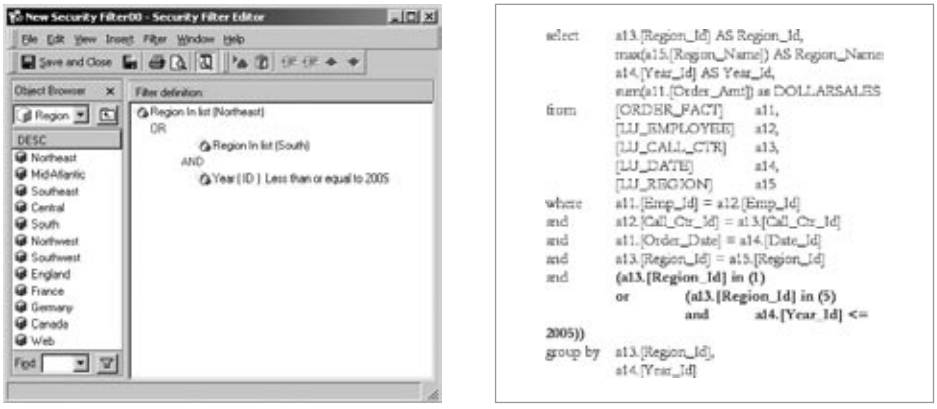


Figure 8.6 A filter applied to Northeast Manager; the resulting SQL of a report includes the filter condition.

Security filter definitions may also specify Top and Bottom range attributes. A Top range attribute specifies the highest level of detail that the security filter allows the user to view. If a Top level is specified, the security filter expression is NOT raised to any level above the Top level. A Bottom range attribute specifies the lowest level of detail that the security filter allows the user to view. If this is not specified, the security filter can view every level lower than the specified top range attribute that otherwise meets the filter expression criteria.

Many companies use hand-coded SQL queries in their MicroStrategy BI applications to retrieve and distribute select information to groups of business users. The difficulty of ensuring appropriate data security on these freeform SQL queries is significant. MicroStrategy addresses this difficulty by embedding security filters within freeform SQL queries. Though these SQL queries are hand-coded and static, they can include a "wildcard" that will dynamically insert the appropriate user security condition at run time while leaving the rest of the custom SQL query unchanged.

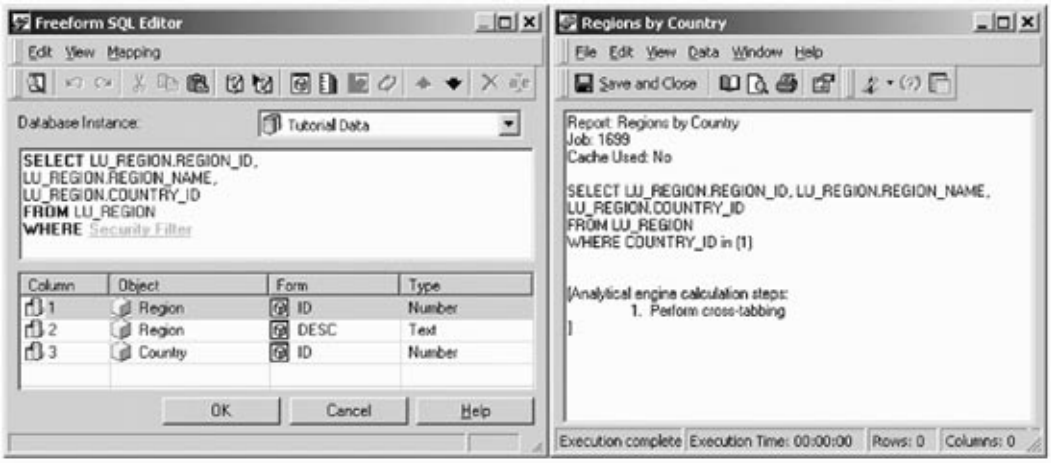


Figure 8.7 Security filters are embedded into freeform SQL queries at run-time.

MicroStrategy also provides a user login condition that incorporates individual user logins as a condition in the SQL query. This ability integrates user data access based upon security tables inside the data warehouse with security set up inside the MicroStrategy metadata. The user login prompt can be set to apply to all users of a BI application or to specific users or user groups. Even more fine-grained control is possible by including the user login condition on a report-by-report basis. For example, all reports run by a user "Meredith Roe", would include an additional filtering clause in the SQL query with her login, "mroe", dynamically inserted at run-time.

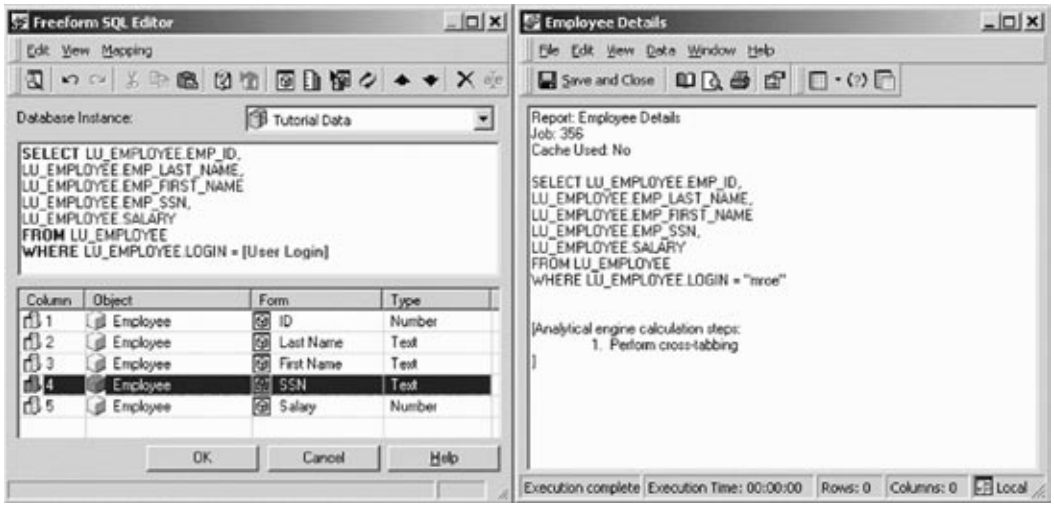


Figure 8.8 The SQL engine dynamically inserts the user id of the authenticated user.

Organizing Users into User Groups and Security Roles

With all of the granular controls for user authorization described above, administrators can easily be overwhelmed by the task of assigning security parameters for every individual. While offering maximum flexibility with full control at the individual level, MicroStrategy's administration architecture also employs a far more powerful and scalable mechanism to organize security profiles into group- and role-based user assignments. All elemental building blocks

of User Authorization – Application Functionality Privileges, Object Access Permissions, and Data Access Security – are first assigned to a series of user groups and security roles. Users are then assigned to any number of groups and roles, and the MicroStrategy architecture dynamically calculates privileges, permissions, and security filters to create an aggregate user profile for each user upon login.

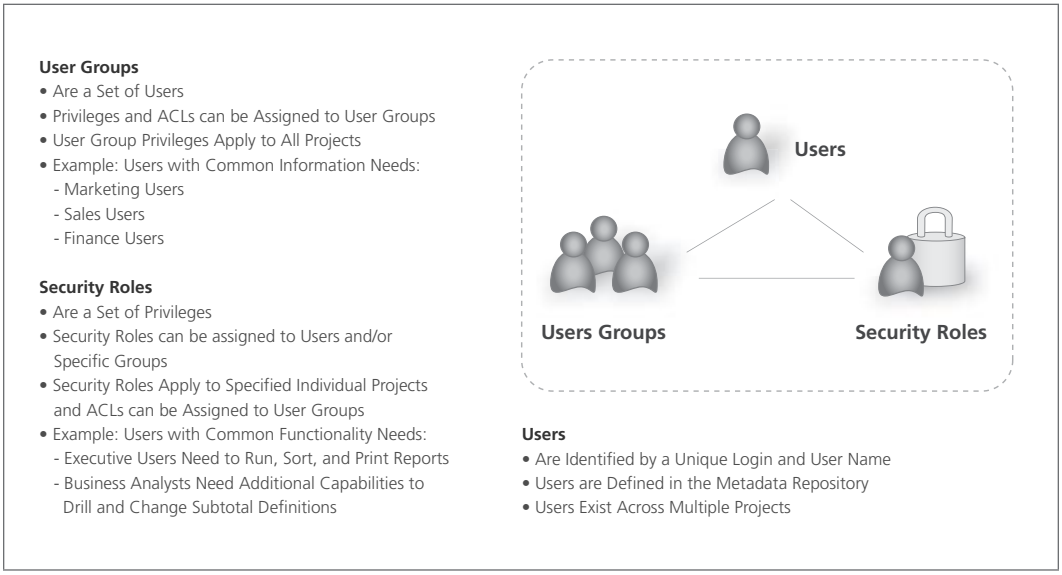


Figure 8.9 All three dimensions of user authorization (application functionality privileges, object access permissions, and data access security) may be defined at the user, group or security role level.

Security is managed in MicroStrategy in two ways.

1. Intuitive graphical interfaces make setting up and administering security an easy task. Editors and wizards within MicroStrategy Desktop provide a non-programming environment to change security definitions quickly from a single location. Users can also be imported from external security repositories.
2. Consistent with MicroStrategy's emphasis on maximum flexibility and scalability, text-based commands can easily make mass changes to users, groups, roles, and security profiles. These commands can be generated from third-party tools to automate common tasks and reduce administrative overhead.

The end result is that every system user has a personalized user profile that governs all privileges across all BI applications and reports. Each user sees only pertinent data, and interacts with the data in a controlled way, regardless of who created the report, and how the report is defined.

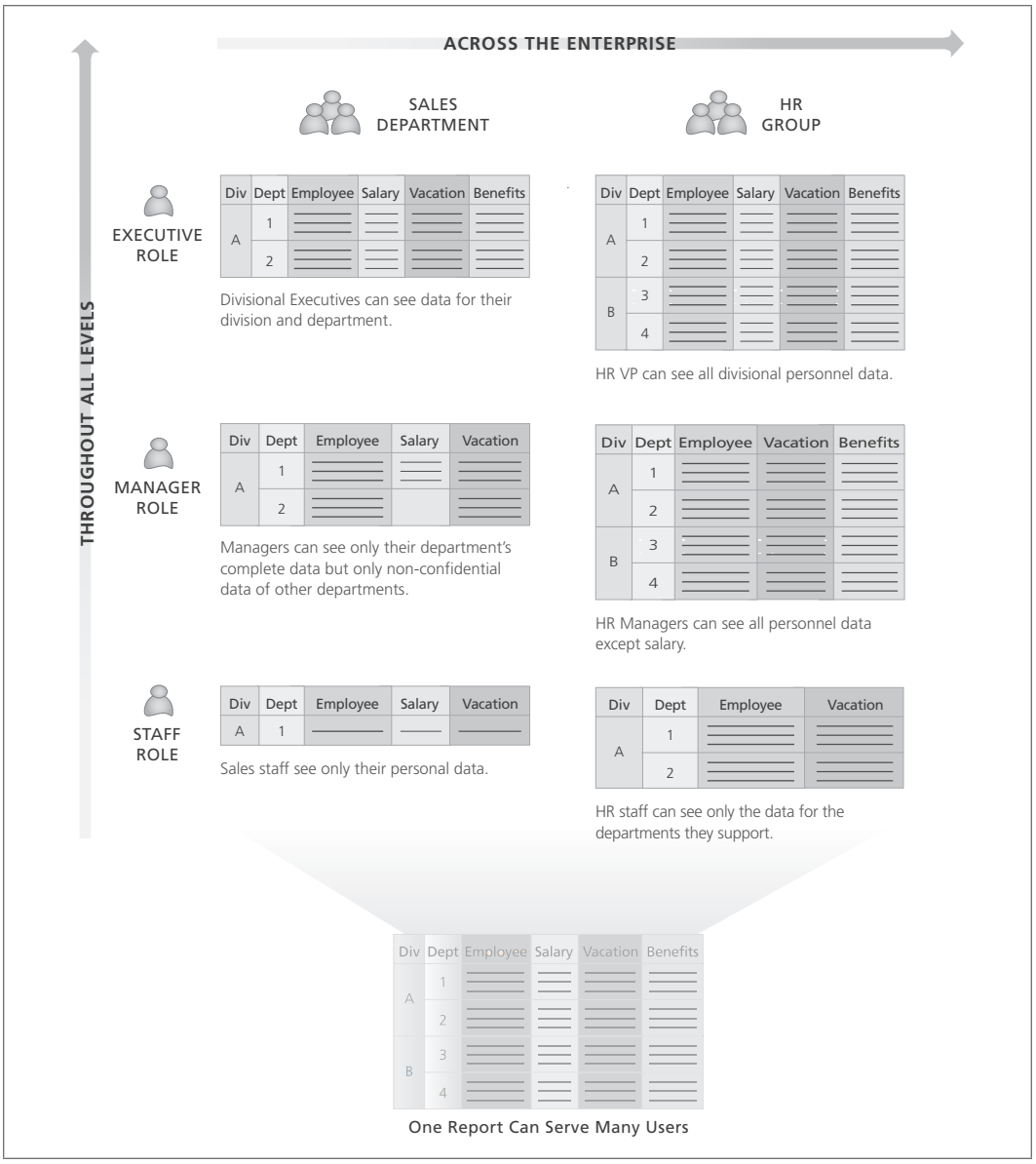


Figure 8.10 Each MicroStrategy user is assigned to any number of user groups and security roles, which combine to define each user's unique security profile. The profile is applied to every report, ensuring that all data is fully secured regardless of who developed the report or who is running it.

8.4 Multi-Tier Web Architecture and Transmission Security

As a Web-based reporting, analysis, and monitoring platform, MicroStrategy supports the most stringent Internet security requirements in two ways:

1. MicroStrategy conforms to the most widely accepted security architecture standards by using a third generation multi-tier architecture as its foundation.
2. MicroStrategy incorporates several data encryption algorithms that ensure the secure transmission of data to all users accessing MicroStrategy BI applications.

Third Generation Multi-tier Architecture

Companies are deploying more and more applications over the Web, making security requirements much more stringent. MicroStrategy addresses these requirements with an elegant architectural design that is optimized for high performance and high scalability while providing secure information to all BI constituents. Five architectural tenets ensure the integrity of the data in the BI system.

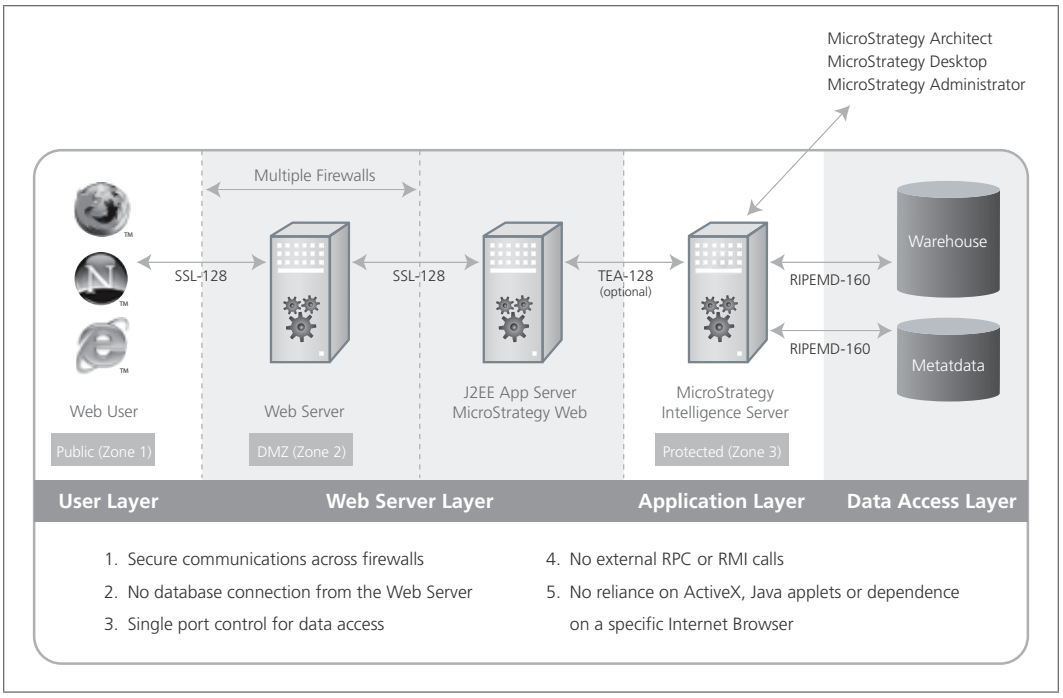


Figure 8.11 Communications between the components of the MicroStrategy platform form multiple protected zones.

1. Secure communications across firewalls
   The MicroStrategy platform is normally installed on more than one server to distribute the BI workload. Secure communication across these servers is often governed by layers of firewalls constructed into Demilitarized Zones (DMZ). Using multiple firewalls, two distinct DMZ's are created with one DMZ protecting the Web server and the second DMZ securing the infrastructure of the data source and Intelligence Server.

2. No database connection from the Web Server
   An effective DMZ is characterized not only by the mere presence of firewalls. It is equally important that the architectural component that accesses the database should reside behind a firewall. MicroStrategy Intelligence Server is the core of MicroStrategy's BI platform, and is the only component that accesses the database. It resides between two firewalls in the same way that the Web Server resides between two firewalls. Only in this configuration is a hacker who gains access to the Web server prevented from accessing the database.

3. Single port control for data access
   Firewalls protect corporate information assets by limiting which application has access rights to certain computer network ports. To take full advantage of this protection, the Web-based application must allow for granular port access control. MicroStrategy's Web architecture allows administrators to configure which port is used for data access and for inter-server communication. Connections to other ports can be disallowed by the firewall, thus minimizing exposure.

4. No external Remote Procedure Calls (RPC) or Remote Method Invocation (RMI) calls
   RPC and RMI calls are hazardous because they allow hackers to access and control remote and distributed computer processes. These calls often allow anonymous access through separate, open ports in the firewall. MicroStrategy Web uses only XML to communicate with Intelligence Server, eliminating the need for RPC or RMI calls completely.

5. No reliance on ActiveX or Java applets or dependence on a specific Internet Browser
   Business partners and customers who access an extranet business intelligence application usually have to cross the safety of their own corporate firewall, which often prevents all incoming ActiveX or Java applets as a standard security procedure. Unlike other products that mandate the use of applets, downloaded compiled code or a specific Internet browser, MicroStrategy's Web uses HTML and DHTML, and is designed to be browser-independent. This true zero footprint Web interface provides maximum end user functionality and flexibility without limiting Web deployment and security options.

Data Encryption and Transmission

As corporate business information is transmitted over the Internet, security of the data itself becomes important. MicroStrategy's BI platform can be set to encrypt data as it is being transmitted across the network or while it is being stored. MicroStrategy uses the following encryption protocols:

1. Secure Socket Layers (SSL)
   Communication between Web browsers and the Web server can be encrypted using SSL. This uses a public and private key encryption system from Rivest-Shamir-Adleman (RSA), and includes the use of digital certificates. As SSL technology continues to improve, MicroStrategy's BI platform seamlessly takes advantage of better and stronger security.

2. Tiny Encryption Algorithm (TEA)
   Report results that are sent from Intelligence Server to MicroStrategy Web can be encrypted using TEA. This is a 2-way encryption algorithm that uses a 64-bit block cipher with a 128-bit key length.

3. RACE Integrity Primitives Evaluation MD-160 (RIPEMD-160)
   Passwords that need to be stored are encrypted using a one-way cryptographic hash function called RIPEMD-160. This encryption occurs before the password is transmitted across the network, ensuring that network sniffers have no way of obtaining clear-text passwords.

8.5 Enterprise-Class Security at Every Level of the BI Architecture

The MicroStrategy BI platform's industrial strength security architecture is built to meet the highest security requirements of any organization. Seamless integration with Windows, LDAP, databases, and other third-party systems ensures that only appropriately authenticated users are granted access to the BI system. Combining data access security with application functionality privileges and object access permissions provides granular, cell-level data security that is personalized for each individual, user role or user group. This personalized security of corporate information combines with a multi-tier Web architecture, robust encryption functionality, and a true zero-footprint client in order to offer a layered approach to protecting sensitive data.

9

Reliability and Fault Tolerance

High availability and reliable high performance are critical requirements for companies implementing enterprise-class business intelligence systems for day-to-day business decision making. Companies must be assured that their implementation is capable of delivering 24x7 availability, that it can scale easily, and that it will perform reliably under heavy loads.

MicroStrategy customers can guarantee high availability via clustering, which is the ability to group multiple servers, called nodes, into one functional unit. If any node fails, users and jobs are transparently routed to available nodes in the cluster in a manner that avoids the dropping or duplication of job processing. This dynamic rerouting is called "failover." MicroStrategy's clustering and failover capabilities make it resilient to failures of particular nodes.

9.1 Stateful and Stateless Architectures

Software applications can be designed to be "stateful" or "stateless." In a stateful system, significant information about the user and the service is kept in memory (e.g., user session information, cached data, and history of recent user interactions). Stateless systems do not maintain such data, and each interaction with the software has no memory of prior interactions. Such systems use less memory, but performance may be lower since each request needs to rediscover information that is kept in memory in a stateful system.

Since critical components require higher performance, MicroStrategy designed its most critical components, Intelligence Server and Narrowcast Server, to be stateful. Less critical MicroStrategy components are designed to be as stateless as possible. This dual approach is illustrated below:

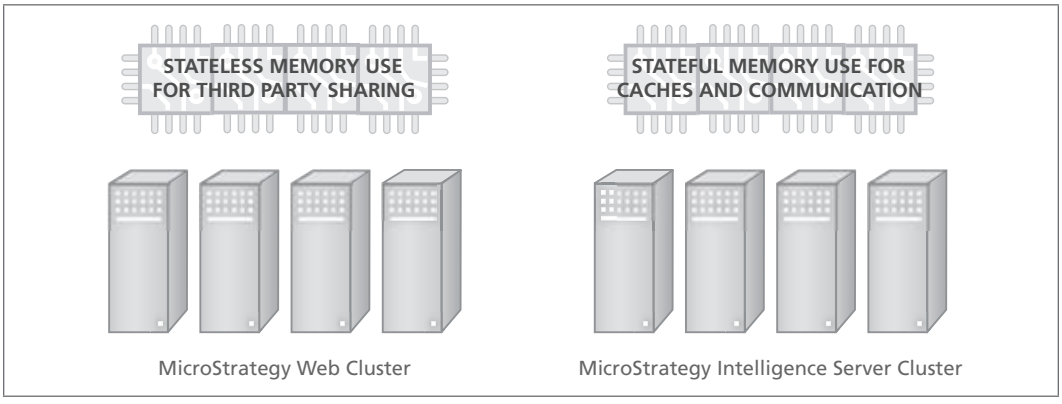


Figure 9.1 The MicroStrategy BI platform is designed to be both stateful and stateless, to work with third-party expertise where appropriate, and to provide MicroStrategy clustering capability only where MicroStrategy has core expertise.

MicroStrategy interacts with third-party components, such as the application server, Web server, file server, and database server, in a standard, stateless manner; these components can work seamlessly with third-party clustering mechanisms. MicroStrategy servers are designed to work with server system management software so that, if a server running MicroStrategy Intelligence Server stops, the load can be taken on by the other nodes in the cluster.

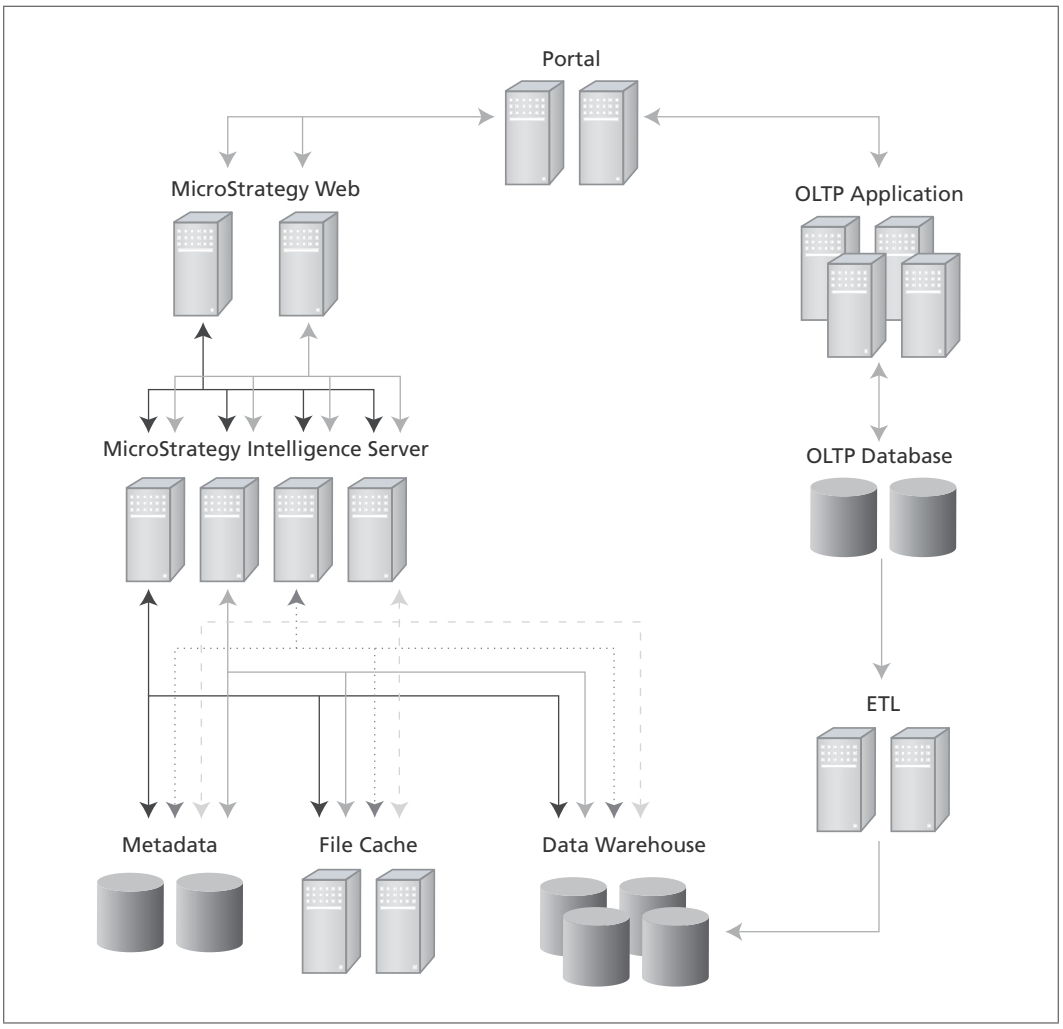


Figure 9.2 A typical third-party enterprise system management "console view" of a system using MicroStrategy.

In the stateful portions of this system architecture, implementing effective clustering and failover becomes a complex problem due to cached session data. In order for failovers to be transparent to the user, the data that is cached must be available to a node that starts to service the jobs of a failed node. Solutions from third parties cannot effectively cluster Intelligence Server and Narrowcast Server since they would be unaware of the cached data in memory. Therefore, MicroStrategy offers a clustering solution that makes intelligent use of this data to provide high availability.

Clustering business intelligence servers is a considerable technological achievement. At its core, MicroStrategy offers three key architectural characteristics that make clustering possible:

1. A completely separable and modular set of components
2. Shared, unified metadata
3. Effective intra-cluster communication

9.2 Modular Components Can Be Kept Together or Separated

The MicroStrategy architecture encourages enterprises to optimize their BI environments at all stages of the project lifecycle. When the project is small, all components (data warehouse, metadata, file cache, server, and Web server) can reside on a single server. As BI requirements shift towards critical high availability, each component can be separated from the rest, placed on optimized server resources, and clustered to provide complete reliability and fault tolerance.

9.3 Shared, Unified Metadata

MicroStrategy's customers have always required the best and most efficient use of metadata, especially when supporting the industry's largest user populations and most robust BI applications. MicroStrategy has long maintained a single shared metadata for all servers, projects, users, and reports. This shared metadata can be separated from the rest of the architectural components, and placed on an optimized server. All nodes of the cluster can point to this central metadata repository, establishing a single version of the truth for all users.

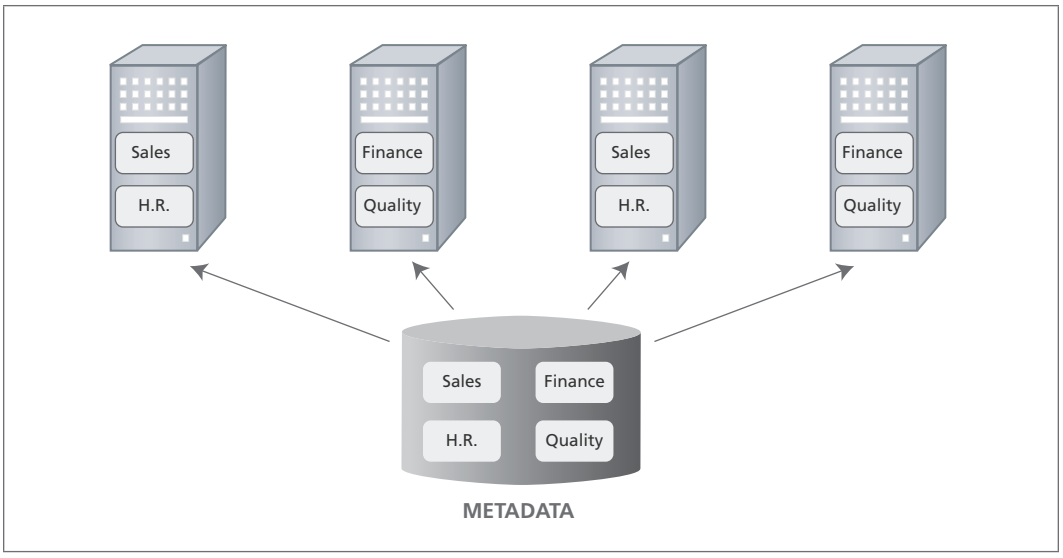


Figure 9.3 A single shared and optimized metadata for all nodes in a cluster.

9.4 Communication Between Cluster Nodes

A key requirement for efficient clustering is that the nodes of the cluster be able to share and synchronize information on an as-needed and scheduled basis. This capability is achieved with cluster-node communication. The architectural philosophy behind this functionality is that it is far more efficient to share and synchronize information between cluster nodes, consuming a few milliseconds of time and a few bytes of network bandwidth, than to duplicate metadata or report caches.

The Intelligence Server cluster is an all-active peer-to-peer configuration in which:

- Each node is aware of data available to other nodes
- Each node shares and can access data across all nodes

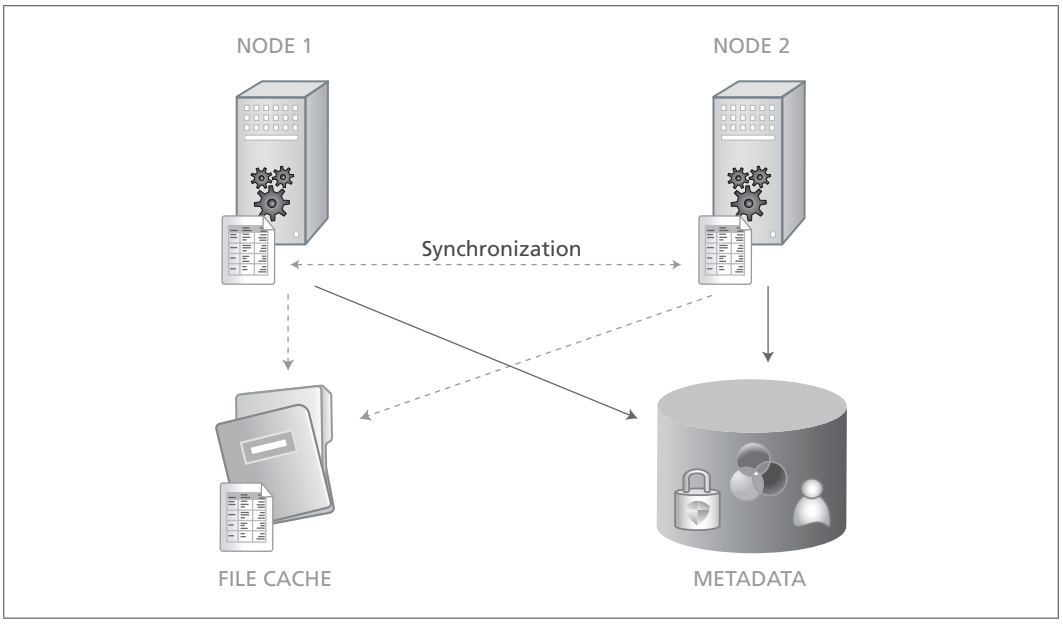


Figure 9.4 All metadata and report file cache data is available to all nodes.

Nodes maintain their own session caches and report caches. The nodes of an Intelligence Server cluster communicate every time the in-memory or the file-based report cache is changed by any node. Moreover, only the exact references to the updates are communicated, which simplifies the message, and reduces the size, complexity, and the time taken to transmit and process the updates. With the updates, each node can access cached results from all another nodes. In order to maximize network efficiency and avoid data replication, no actual report data is transmitted.

9.5 MicroStrategy Clustering Capabilities

MicroStrategy's modular components, unified metadata, and intra-cluster communication are the foundation of a robust clustering architecture that supports asymmetric clustering and automatic failover for the Intelligence and Narrowcast Servers.

Dynamic Provisioning with Asymmetric Clustering

Organizations deploy many different BI applications to monitor and analyze the performance of an increasing number of business processes. Loading many applications on a single node can consume a relatively large amount of memory. Intelligence Server addresses this problem by supporting asymmetric clustering, in which each Intelligence Server node within a cluster can host a different set of projects. This feature enables further opportunities to optimize the overall BI application, better utilizing available resources and increasing performance through:

- Further optimized resource allocation
- Added load balancing configurations for MicroStrategy Web, MicroStrategy Office, and Narrowcast Server
- Increased agility to react to usage fluctuations
- Enhanced Failover and Fallback Logic

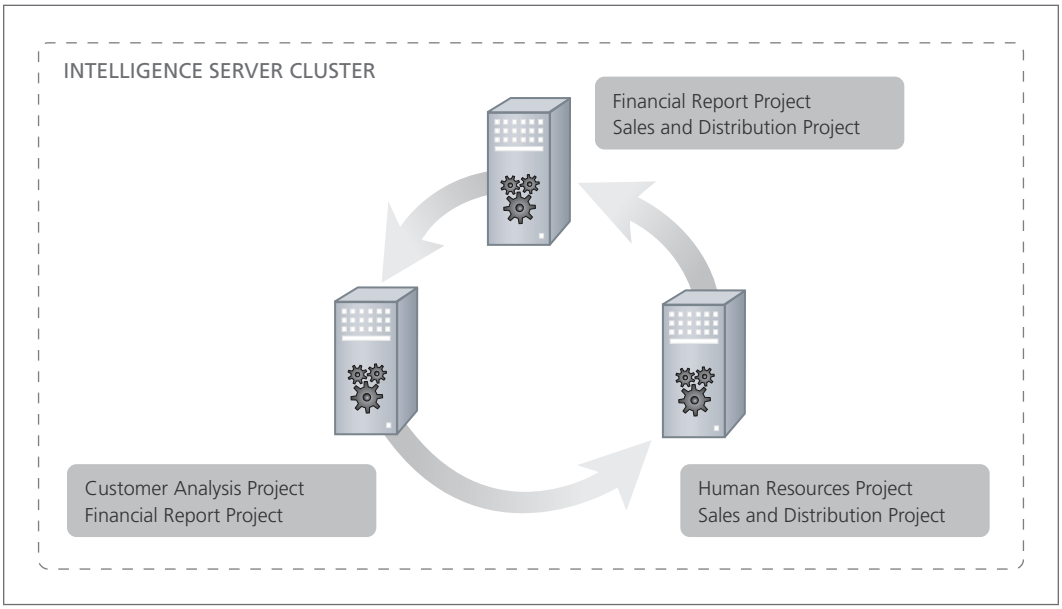


Figure 9.5 An example of asymmetric clustering in Intelligence Server

With Asymmetric Clustering, an organization can systematically distribute projects across different Intelligence Server nodes within a cluster. Factors influencing the distribution include:

- Usage pattern
  - Average: A collection of projects may be more resource-intensive than others. These can be better serviced by having particular nodes service only these projects, while other projects are more distributed.
  - Periodic: A collection of projects may require more resources at certain hours of the day. These projects can be better serviced at those hours by loading them on more nodes while reducing the number of nodes on which other projects operate.

- Business priorities
  - Mission-critical projects that are used by a large number of users can be loaded onto all nodes in the cluster, while projects used by few users can be loaded on a subset of the nodes. This approach not only provides excellent fault tolerance, but also ensures high scalability and performance. Distributing projects across at least 2 nodes ensures that those projects will be available even if one of those nodes becomes unavailable.

Asymmetric clustering provides flexibility in using BI resources, as the number of nodes running a project can be modified to accommodate the requirements of the users.

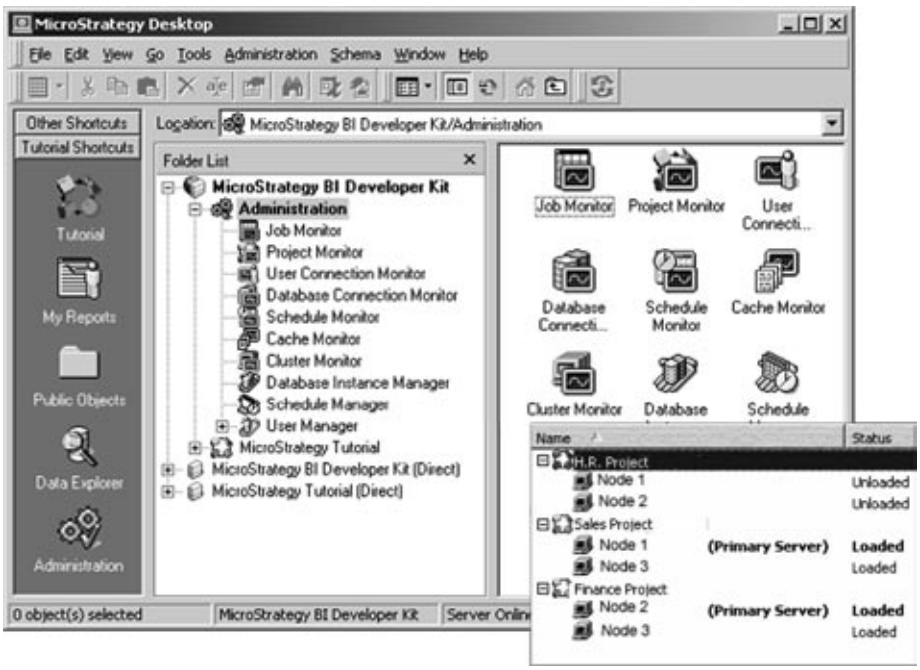


Figure 9.6 Cluster nodes can be administered easily using the Desktop graphical interface.

Clusters are administered through the administrative console in the Desktop interface, and via commands in Command Manager. It is easy to modify the capacity of a growing BI environment because it is simple to add or remove nodes from a cluster. This means that system maintenance and system growth are far more easily accommodated in a cluster than by administering a single machine. Furthermore, the status of all nodes and projects on those nodes in the cluster can be viewed in these interfaces.

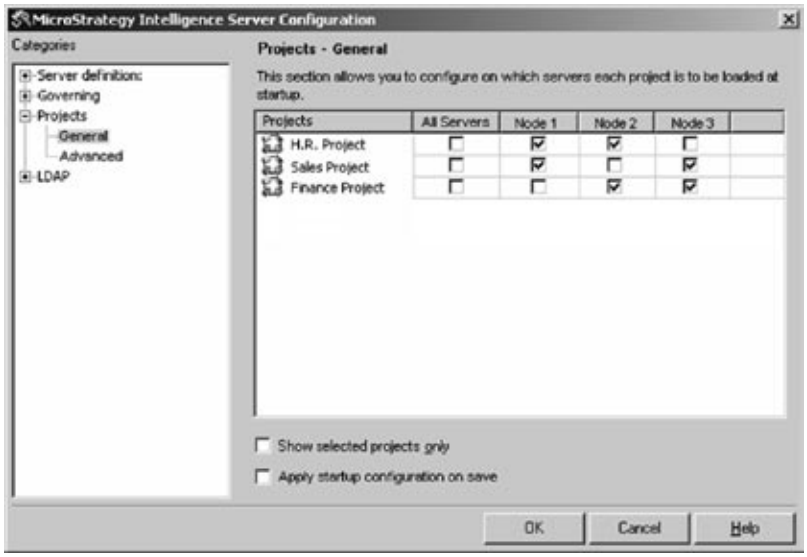


Figure 9.7 A view of the list of projects and their status sorted by the nodes of the cluster.

9.6 Automatic Failover

In the event of a fault in one node, all sessions in related components (MicroStrategy Web, MicroStrategy Office, and MicroStrategy Narrowcast Server) are transferred to another node in the cluster. The failover process is transparent to end users, who will not even realize that a failure occurred. Executing reports from the node that failed are automatically queued ready to be automatically re-executed on the newly-allocated node.

The system administrator has a choice of hot or cold failover, each of which is invisible to the end user.

Sharing the History List for Hot Failover

The Intelligence Server's History List stores report and job messages requested by each individual user. Each node in a cluster maintains its own set of history list reports and messages for users that have previously logged in to that particular node. The communication between nodes transmits the location of all history list data stored in each node. When a user logs into a node, all the relevant history list information from all nodes in the cluster are loaded on that node. In addition, history lists are automatically synchronized across nodes. In case a node fails, another node will take over the processing, and the completed report will run on the other node.

Sharing the Report Caches for Cold Failover

Cold Failover refers to report results being sent to a shared report cache on a file server. If a node goes down, a second node will reference and load the report cache when the user session is transferred. This functionality is supported by a cache lookup table, which is replicated in memory on each node, which maintains the locations of all report caches for a particular project. Each node is aware of the caches on the other nodes, and all cache lookup tables on all nodes are automatically synchronized via cluster broadcasts of lookup table updates.

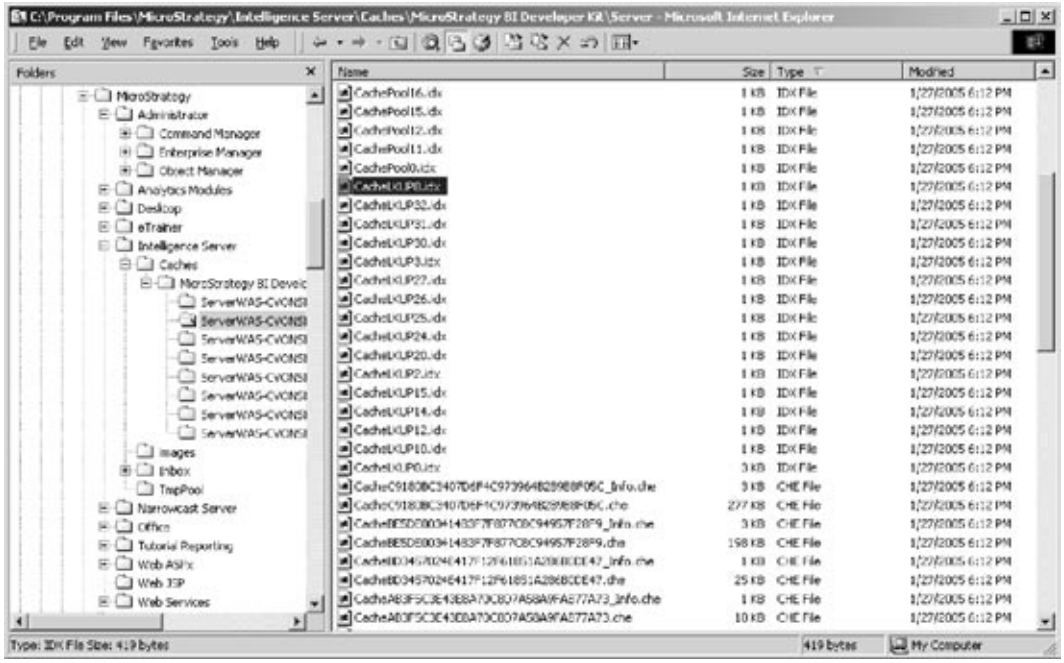


Figure 9.8 Cache lookup tables are maintained so that the shared file cache can be referenced from either node at run-time.

9.7 Clustering for Batch-Processing and Report Distribution

Narrowcast Server's Delivery Engine has built-in redundancy down to the component level, ensuring that delivery service messages are not dropped or duplicated. Using a master-slave clustering architecture and an effective "divide and conquer" algorithm for service execution, Narrowcast Server has built-in failover and load balancing capability at multiple levels:

- Backup Distribution Managers provide failover for Narrowcast Server operations
- Multiple Execution Engines provide workload balancing
- Retry logic provides robust handling of failure situations
- Failover tracking of each message ensures message delivery and avoids duplication

A backup Distribution Manager can be installed on a separate machine. This backup machine continually polls the primary Distribution Manager and automatically takes over if it stops receiving a response. It uses a diagnostic log file to determine which processes were interrupted, and resubmits them.

Multiple Execution Engines can also be implemented. The Distribution Manager balances the load of delivery service requests by assigning them to different Execution Engines. The Execution Engines may be on the same machine or on several different machines to promote scalability.

Narrowcast Server has built-in retry logic for delivery services. If an external resource, such as access to subscription information, becomes unavailable, the Distribution Manager will stop service execution until that resource becomes available again or the service expiration time is reached.

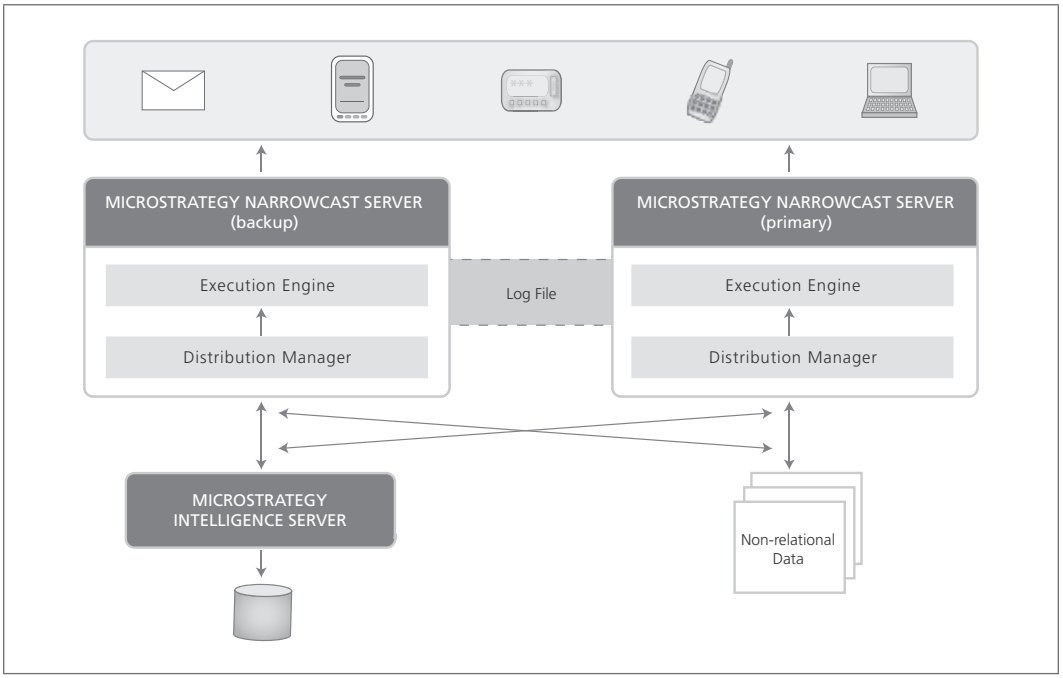


Figure 9.9 In Narrowcast Server, a backup Distribution Manager can be configured to automatically take over if the primary Distribution Manager becomes unavailable.

Narrowcast Server also tracks the execution and distribution of each personalized message. If the system is disrupted in any way, there is seamless recovery from these disruptions. No message is ever dropped or duplicated. In configurations with multiple Execution Engines, each Execution Engine is aware of the delivery services that are allocated to other Execution Engines. If one Execution Engine fails, another Execution Engine will automatically take over the jobs that were not completed by the failed engine.

Adaptation to Changing Conditions

A final aspect of reliability is the ability to perform consistently under varying conditions, such as those resulting from increased or unexpected demands on the business intelligence system. To alleviate potential processing bottlenecks that can occur as demands on the system increase, the platform configuration can be adjusted manually or automatically using one of the following mechanisms:

- Manual adjustment (by the administrator)
- Self-tuning built into the software
- Integration with system management software

The MicroStrategy platform can automatically reallocate computing resources in response to changing demands on the business intelligence system. Either the administrator or system management software can adjust the asymmetric clustering, the number of nodes in a cluster, change the number of open database connections, idle the server or change report execution parameters. There are many other settings that can be used to control the platform's operations and performance at a granular level.

9.8 Summary

MicroStrategy's native clustering capabilities, comprehensive real-time monitoring, and ability to adapt to changing conditions offer the industrial strength reliability and fault tolerance that are necessary to ensure 24x7 availability of the business intelligence system. Clustering at multiple levels in the business intelligence system insulates users from hardware and operating system failures, ensuring that reporting, analysis, information delivery, and transaction execution can be conducted with confidence.

10

Scalability and Performance

Today's companies are collecting ever-increasing amounts of information in their data warehouses. Growing trends of systems integration and automation of business processes provides companies with unprecedented opportunities to amass detailed information and data on all aspects of their operations for comprehensive analysis. Accompanying this increase will also be the number of business intelligence application users. As a result, the BI system supporting the business needs to be scalable. Scalability is the ability to support large numbers of concurrent users accessing and analyzing large sets of data efficiently, and effectively. In addition, it is the ability to implement more hardware resources with proportionally increasing performance. Such a system should be equipped to handle the following issues:

- Increasing numbers of data tables
- Increasing table sizes
- Increasing numbers of active users
- Increasing sophistication of analysis

The MicroStrategy BI platform is the first to meet all of these challenges while maintaining efficient administration and superior performance. Both the enterprise data warehouse and the business intelligence system must be able to handle accelerated data growth so companies can successfully monitor, report, and analyze their operations. As the amount of information increases to the multi-terabyte size, and the number of users potentially reaches millions, it is vital that system performance does not degrade. At the core of the MicroStrategy BI modular architecture is the MicroStrategy Intelligence Server, which is designed from the ground up to meet both today's and tomorrow's business needs. It achieves maximum scalability and performance through characteristics that include:

- Query Optimization
- Multi-level Shared Caching
- Distributable Execution
- 64-bit Technology
- Multi-threaded Processing

Although the initial BI application may not be scoped to reach a large size, the right choice of a scalable BI platform upfront can ensure there are no restrictions to future growth. The MicroStrategy BI platform is the only proven architecture capable of scaling to hundreds of applications, millions of users, and multiple terabytes of data in order to handle all enterprise business intelligence needs.

10.1 Query Optimization

Database vendors invest heavily into building Relational Database Management Systems (RDBMS) that keep up with the proliferation of data. Each RDBMS provides many strategies to deal with this trend. The MicroStrategy BI platform, with its Relational Online Analytical Processing (ROLAP) architecture, uses the RDBMS for much of its processing, making the most of the RDBMS investment. No time-consuming and costly additional processing is needed before business users can access data. MicroStrategy incorporates the following RDBMS optimizations when generating SQL:

- Dynamic indices, statistics, and hints
- Partitioning, clustering, and parallel queries
- Volatile tables, derived tables, and common table expressions
- Materialized views, and parameterized queries

Because of its unique ROLAP approach, MicroStrategy seamlessly works with all of these database-driven strategies for providing quick response time even with massive quantities of data. Any technical advances in the database are seamlessly accessible to the MicroStrategy platform. In addition, the Intelligence Server generates efficient SQL to minimize query response time, and work in tandem with strategies employed by databases. These features are automatically applied for all users but can also be customized to meet unique scenarios. Examples of these features are:

- Multi-pass SQL
- Aggregate Awareness
- Very Large Database (VLDB) properties

Any scalable system must capitalize on the strength of each component of the application. VLDB properties, massively parallel syntax, and intelligent aggregate navigation ensure that the use of the data warehouse – the component in which most companies make the biggest investment – is maximized. Organizations can design and build applications with the assurance that MicroStrategy will continue to leverage each database's strengths to provide optimum performance and scalability.

Multi-pass SQL

One of the key elements to providing analytical sophistication in business intelligence applications is MicroStrategy's ability to generate multi-pass SQL. Multi-pass SQL is required to answer analytical questions that cannot be answered with a single SQL query block. Examples of scenarios that require multi-pass SQL include:

- Set analysis
- Split metrics
- Calculation at different levels of aggregation

Sets allow the definition of complex selections and custom groupings of customers, products, suppliers, or other individual areas of business interest. For example, when data on customers, their recent purchases, and other characteristics related to their transactions is available, sets allow the targeting of customer groups based on specified criteria.



Figure 10.1 Set Analysis using Custom Groups is performed with multi-pass SQL.

The MicroStrategy BI platform enables much more complex analyses. For example, questions such as "Show me sales by region over the last six months, but only for customers who purchased one of the 5 most popular products in 2004, and for regions where vendors have the most reliable shipment times over all time," can be answered. With multi-pass SQL, when sets cannot be calculated together, MicroStrategy breaks them down into manageable components to achieve results quickly. Similarly, queries requiring calculations against different database tables with complex relationships cannot always be resolved with direct queries and require multiple steps. The Dynamic SQL Engine recognizes the structure of the tables, and generates correct SQL to ensure accurate data with fast performance.

Often times, organizations need to compare results between different strata. These calculations involve aggregating data into different levels for cross comparison. Individual performances are compared to total group performances to obtain insight into relative value. For example, stores contributing the highest revenues to their regions are identified as benchmarks, and all other stores are compared against them. When this comparison is done at the percent contribution level, no custom SQL coding is required; the analysis is automatically supported using multi-pass SQL.

Support for these scenarios, especially when combined together, provides a framework for answering significant analytic questions, and creating tremendous value to business users. MicroStrategy generates SQL that performs these multi-pass queries as efficiently as possible for each particular database.

Aggregate Awareness

The MicroStrategy BI platform features intelligent aggregate table navigation. Query performance in many data warehouses is enhanced through the use of aggregate tables or summary tables, which store pre-computed aggregate results. Report queries access information from a summarized set of data rather than the detail level data that would be stored in the fact table, often improving query performance by orders of magnitude. Administrators can build as many aggregate tables as are necessary and useful.

MicroStrategy has used an aggregate-aware SQL-generation engine since its products first became available in 1994. MicroStrategy transparently navigates aggregate tables, dynamically directing queries to summary tables when doing so is optimal, without the user having to specify use of the table. Once these tables are included in the application, Intelligence Server automatically evaluates all the potential options for generating the results, and intelligently chooses the best aggregate tables for the query each time SQL is generated.

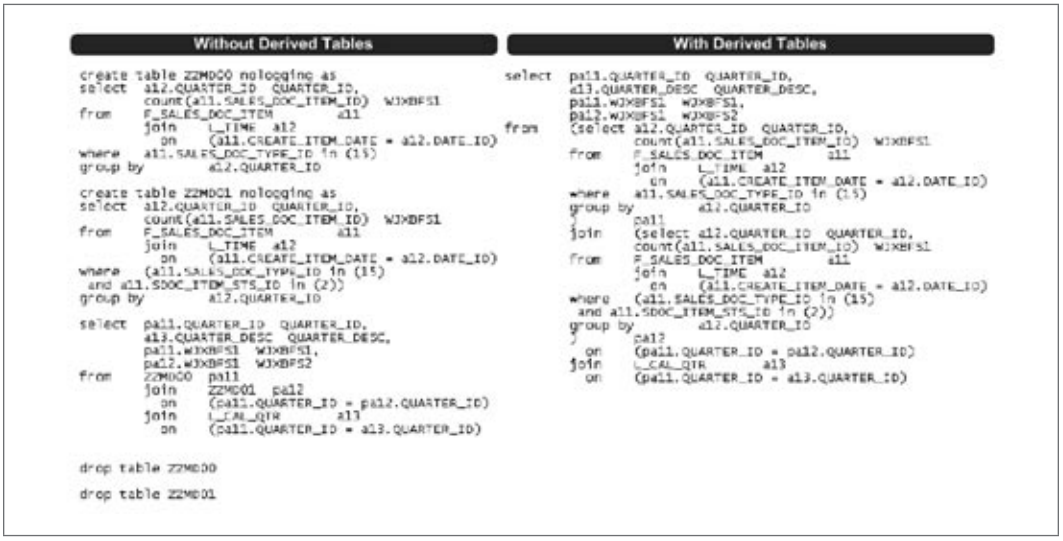
Very Large Database (VLDB) Parameters

Each major relational database has specific syntax to make use of its query optimization features. Small changes to the SQL can guide the database optimizer, and provide a dramatic improvement in query performance.

The MicroStrategy BI platform automatically ensures that each query leverages the high-end capabilities of the database through VLDB properties for the databases. VLDB properties are parameters that optimize the SQL generated for performance. Intelligence Server is automatically configured to use the right set of VLDB properties for each major database when that particular database is selected. Using these out-of-the-box VLDB properties, Intelligence Server adapts to the existing environment, and capitalizes on the inherent strengths and native optimizations of each database platform. Business users are not required to understand any usage rules within the business intelligence system or within the database to benefit from VLDB settings.

For example, databases that support derived tables can provide orders of magnitude increases in performance over generic SQL. Rather than generate generic SQL, Intelligence Server generates SQL that uses derived tables whenever possible. The following SQL sample of a report shows two different SQL statements generated when the derived table switch is toggled. In this case, the response time was approximately 190 % faster when derived tables were used.

As each BI implementation is unique, the VLDB properties can be manually adjusted to suit a particular database schema, metric calculation, or report. Over 40 properties in 11 categories are customizable to produce highly efficient SQL queries.



Without Derived Tables

```
create table ZZMD00 nologging as
select  a12.QUARTER_ID  QUARTER_ID,
        count(a11.SALES_DOC_ITEM_ID)  WJXBFS1
from    F_SALES_DOC_ITEM        a11
        join    L_TIME  a12
          on    (a11.CREATE_ITEM_DATE = a12.DATE_ID)
where   a11.SALES_DOC_TYPE_ID in (15)
group by        a12.QUARTER_ID

create table ZZMD01 nologging as
select  a12.QUARTER_ID  QUARTER_ID,
        count(a11.SALES_DOC_ITEM_ID)  WJXBFS1
from    F_SALES_DOC_ITEM        a11
        join    L_TIME  a12
          on    (a11.CREATE_ITEM_DATE = a12.DATE_ID)
where   (a11.SALES_DOC_TYPE_ID in (15)
 and a11.SDOC_ITEM_STS_ID in (2))
group by        a12.QUARTER_ID

select  pa11.QUARTER_ID  QUARTER_ID,
        a13.QUARTER_DESC  QUARTER_DESC,
        pa11.WJXBFS1  WJXBFS1,
        pa12.WJXBFS1  WJXBFS2
from    ZZMD00  pa11
        join    ZZMD01  pa12
          on    (pa12.QUARTER_ID = pa12.QUARTER_ID)
        join    L_CAL_QTR       a13
          on    (pa11.QUARTER_ID = a13.QUARTER_ID)

drop table ZZMD00

drop table ZZMD01
```

With Derived Tables

```
select  pa11.QUARTER_ID  QUARTER_ID,
        a13.QUARTER_DESC  QUARTER_DESC,
        pa11.WJXBFS1  WJXBFS1,
        pa12.WJXBFS1  WJXBFS2
from    (select a12.QUARTER_ID  QUARTER_ID,
                count(a11.SALES_DOC_ITEM_ID)  WJXBFS1
        from    F_SALES_DOC_ITEM        a11
                join    L_TIME  a12
                  on    (a11.CREATE_ITEM_DATE = a12.DATE_ID)
        where   a11.SALES_DOC_TYPE_ID in (15)
        group by        a12.QUARTER_ID
        )       pa11
        join    (select a12.QUARTER_ID  QUARTER_ID,
                count(a11.SALES_DOC_ITEM_ID)  WJXBFS1
        from    F_SALES_DOC_ITEM        a11
                join    L_TIME  a12
                  on    (a11.CREATE_ITEM_DATE = a12.DATE_ID)
        where   (a11.SALES_DOC_TYPE_ID in (15)
         and a11.SDOC_ITEM_STS_ID in (2))
        group by        a12.QUARTER_ID
        )       pa12
          on    (pa11.QUARTER_ID = pa12.QUARTER_ID)
        join    L_CAL_QTR       a13
          on    (pa11.QUARTER_ID = a13.QUARTER_ID)
```

Figure 10.2 Different SQL statements generated when the 'derived table' switch is turned on.

VLDB Property Levels

VLDB properties can be configured at different levels in the MicroStrategy BI environment. A system administrator can customize the global and application settings, while a report designer can tweak individual metrics, templates, and reports. In the following diagram, the VLDB properties set at the report level override the same properties set at higher levels.

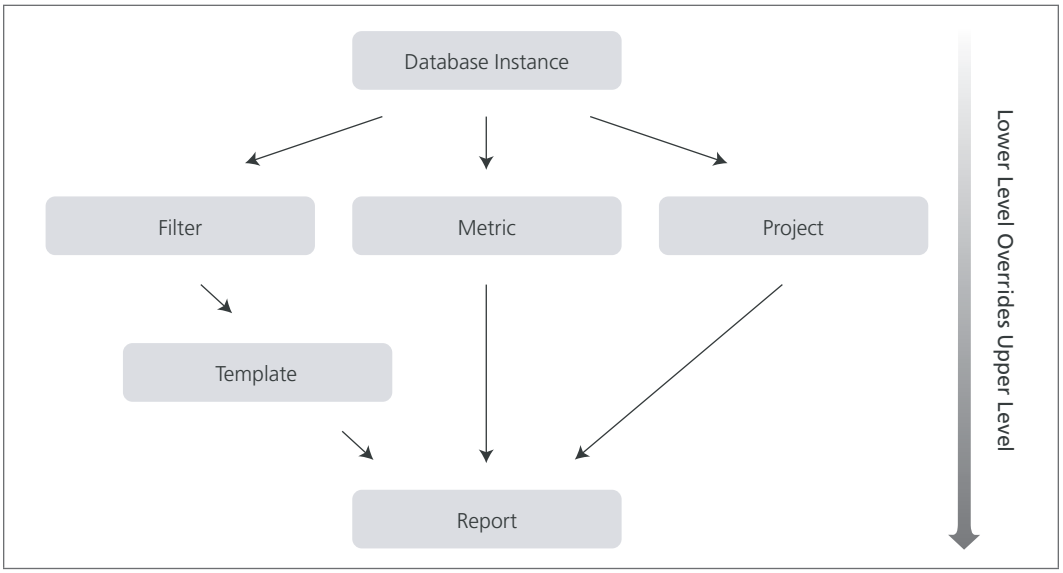


Figure 10.3 VLDB Properties Hierarchy determines which VLDB properties to apply to a query.

Types of VLDB Properties

VLDB properties at each level are defined in a simple graphical interface in the Windows development environment. A SQL preview shows the effects of changing a setting. The VLDB properties fall mainly into the following types:

- Query optimization
- Fallback optimization
- Join guidance
- Parallel execution guidance
- Database tuning optimizations

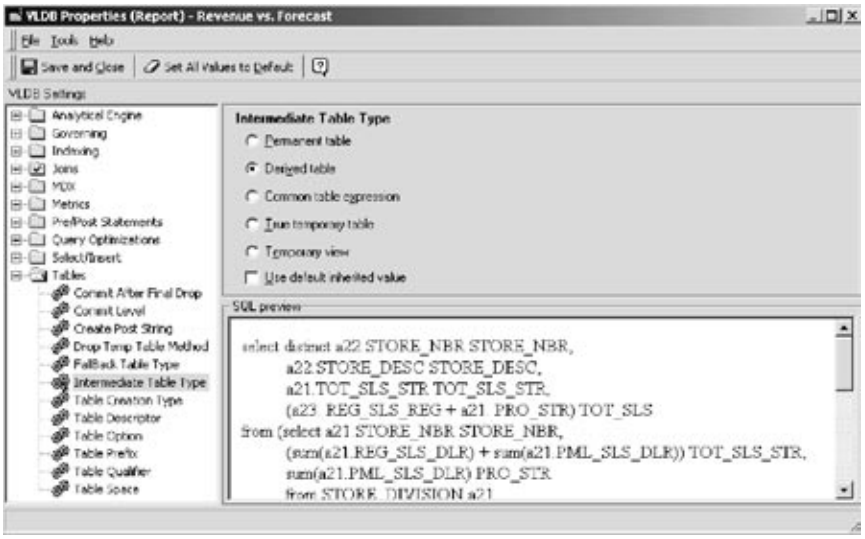


Figure 10.4 VLDB properties that can be set for a specific report.

SQL Query Optimization

Query optimization properties generate SQL for the best possible query performance by adjusting the Data Definition Language (DDL), and Data Manipulation Language (DML) automatically for each major database. Even then, VLDB properties can be customized to match unique reporting scenarios. For example, if using derived tables is not feasible, MicroStrategy can utilize Common Table Expressions instead. Similarly, if joins to attribute descriptors impede performance, MicroStrategy can delay these joins until the final SQL pass. The latter can reduce query execution times by 90% or more.

In addition, an experienced Report Designer has the opportunity to further improve query performance by orders of magnitude with custom Optimizer hints. For example, the Oracle optimizer uses hints that can be customized per report.

SQL Query Optimization Property	Description
SQL Hints	Provides guidance to the optimizer on how best to execute a query
Intermediate Table Type	Provides guidance on how intermediate data is stored. Storage types include derived tables, common table expressions, volatile tables, and global temporary tables.
WHERE clause driving table	Provides guidance on which tables to use in WHERE clause qualifications. This setting is used as an optimization for databases that perform better when columns from smaller tables are used in the WHERE clause.
Additional Final Pass	Provides guidance on when to join back to lookup tables for attribute descriptions. This setting is used in situations where SQL performance can be enhanced by isolating the retrieval of metric calculations from attribute text columns.
Separate COUNT DISTINCT	Provides guidance on issuing COUNT DISTINCT. In certain environments, separating COUNT DISTINCT into isolated SQL statements improves performance by reducing concurrent resource requirements.
UNION Multiple Inserts	Provides guidance on use of UNION clauses rather than multiple separate intermediate tables with INSERT clauses. In certain environments, queries against partitioned tables using UNION clauses can improve performance.

Figure 10.5 Examples of SQL query optimization VLDB properties.

Fallback Optimization

In addition, MicroStrategy VLDB properties provide fallback mechanisms for query optimizations that are used as an alternative to the default setting. For example, if a particular query cannot be executed using derived tables, VLDB properties will reconfigure the generated SQL to alternate intermediate table syntax to ensure that the query returns correct results.

Fallback Optimization Property	Description
Sub-Query Type	Provides guidance on sub-queries syntax. Controls whether EXISTS IN statements or intermediate tables are used when performing sub-queries.
Fallback Table Type	Provides guidance on the intermediate table SQL syntax used. If the primary intermediate table type is unavailable in a database, either permanent or true temporary table syntax will be used.

Figure 10.6 Examples of fallback optimization VLDB properties.

Join Guidance

Correct joins are essential for accurate results and can also impact query performance. MicroStrategy automatically generates optimal SQL with the appropriate joins for data accuracy and query performance. Custom configuration of joins may be required for particular user requests. For example, the sequence in which tables are joined can impact query performance depending on the database. VLDB properties can adjust this sequence for maximum performance.

Join Guidance Property	Description
Lookup Table Join Order	Provides guidance on the sequence in which database lookup tables are joined. This setting is used as an optimization in scenarios in which a database optimizer may not be up-to-date.
From Clause Order	Provides guidance to the sequence in which database tables are used in the FROM clause of SQL. This setting is used as an optimization in scenarios in which a database optimizer may not be up-to-date.

Figure 10.7 Examples of join guidance VLDB properties.

Parallel Execution Guidance

As the volume of data in an enterprise data warehouse increases, partitioning the data provides significant improvements in query performance by distributing the workload between multiple resources in parallel. By dividing an SQL statement among multiple processes, a database can run the statement more quickly than if a single process ran it.

As part of this functionality, high end databases allow tweaks to be added into SQL for additional guidance. For example, when creating tables in a partitioned database, the partitioning key is used to determine the partition on which each row of data is stored. When performing join operations, MicroStrategy uses the partitioning key to improve performance.

Queries with intermediate tables may also benefit from indexes on these tables. In MicroStrategy, most intermediate tables will either be joined on the attribute columns back to fact tables or serve as temporary storage for metrics that will be joined with other intermediate tables to produce a final result. Creating an index on the attribute columns may improve performance, since those columns will usually be involved in a join later in the report SQL.

Parallel Execution Guidance Property	Description
Intermediate Table Index	Partitioning keys can be turned off altogether using the Intermediate Table Index VLDB property. When using this setting with a partitioned database, the database will distribute data according to the first column in the table.
Allow Index on Metric	By default, the partitioning key is created on all attribute ID columns that are used in the intermediate table. Fact columns can also be included in the partitioning key, using the Allow index on metric VLDB property. Depending on joins required, some queries may benefit from distributing data by the fact columns as well as the attribute ID columns.
Max Columns in Index	The application designer can control the size of the partitioning key through the Max columns in Index VLDB property.

Figure 10.8 Examples of parallel execution guidance VLDB Properties.

Database Tuning Optimizations

In order for a database to calculate the most efficient SQL execution plan, optimization engines require up-to-date information especially when new objects, such as intermediate tables, are created. Using VLDB settings, MicroStrategy Intelligence Server can insert complete or partial SQL statements throughout the dynamically generated SQL. For example, the Table Post Statement VLDB setting can be used to update optimizer statistics on new intermediate tables and indexes.

Tuning DB Optimization Engines VLDB Property	Description
Pre/Post Statements	Triggers updates to optimizer statistics when used with database-specific SQL syntax. For example: "Collect Statistics", and "Runstats."

Figure 10.9 Example of VLDB Properties to tune database optimizers.

String Insertion

Besides VLDB properties that automatically alter the overall structure of the SQL statements, MicroStrategy Intelligence Server also provides a multitude of manual entry VLDB settings. These VLDB properties control string syntax used in SQL queries generated by the Intelligence Server. A report designer can further optimize SQL for their specific environment using these string insertion settings. The following illustrates example locations for string insertion for Oracle. Each potential location is highlighted in bold.

```
[Report Pre Statement]

[Table Pre Statement]

 CREATE [Table Qualifier] TABLE [Table Descriptor] [Table Prefix]<table_name>
 [Table Option] [Table Space] AS
 SELECT [SQL Hint] <column_expressions>
 FROM <table_list>
 WHERE <joins_and_filter_expressions>
 GROUP BY <column_expressions>
 [Insert Post Statement]

 CREATE [Index Qualifier] INDEX [Index Prefix]<index_name> ON
 <column_expressions> [Index Post String]

[Table PostStatement]

SELECT [SQL Hint] <column_expressions>
FROM <table_list>
WHERE <joins_and_filter_expressions>
GROUP BY <column_expressions>

[Report PostStatement]
```

Figure 10.10 Example of string insertion VLDB locations.

VLDB Properties Updates

When database vendors add new functionality to their databases, administrators can easily update the VLDB properties. MicroStrategy provides new VLDB property files, along with new releases when necessary, that reflect the added functionality. Administrators simply update their properties using the new file and they are ready to go.

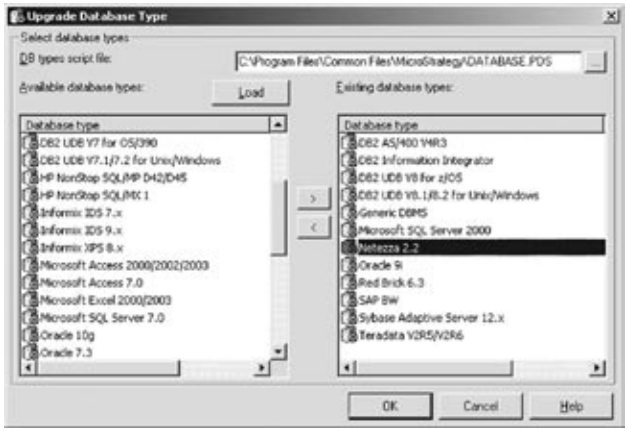


Figure 10.11 Upgrading VLDB properties via a simple dialog box.

10.2 Multi-Level Shared Caching

With other BI products, query performance decreases as the number of users on the system increases. When Intelligence Server caching is enabled, and since cache tuning is based on user activity, performance either improves or remains constant. More users on the system provide the Intelligence Server with a more accurate profile of the most popular reports and data. This allows Intelligence Server to more accurately cache the right data closest to end-users for maximum performance.

Some BI products attempt to implement caching by simply storing data at a single level of the middle tier, thus missing most caching benefits. MicroStrategy Intelligence Servers implements a truly efficient self-tuning caching strategy. The foundation of caching within the MicroStrategy BI platform is to implement caches at key points within the overall query flow, taking into consideration various usage scenarios, and user behavior, to provide the most improvement.

A typical user may log into MicroStrategy, search for a report, build a new report, answer a prompt, run the report, conduct further analysis on the result set, and save the report before logging out. Within this query flow, Intelligence Server caches specific data in the following crucial areas:

- Metadata Object Caching
- Attribute Element Caching
- Database Connection Caching
- Report and Report XML Caching

The diagram below shows how response time dramatically decreases using strategic cache implementation within the Intelligence Server, even as the number of users increases.

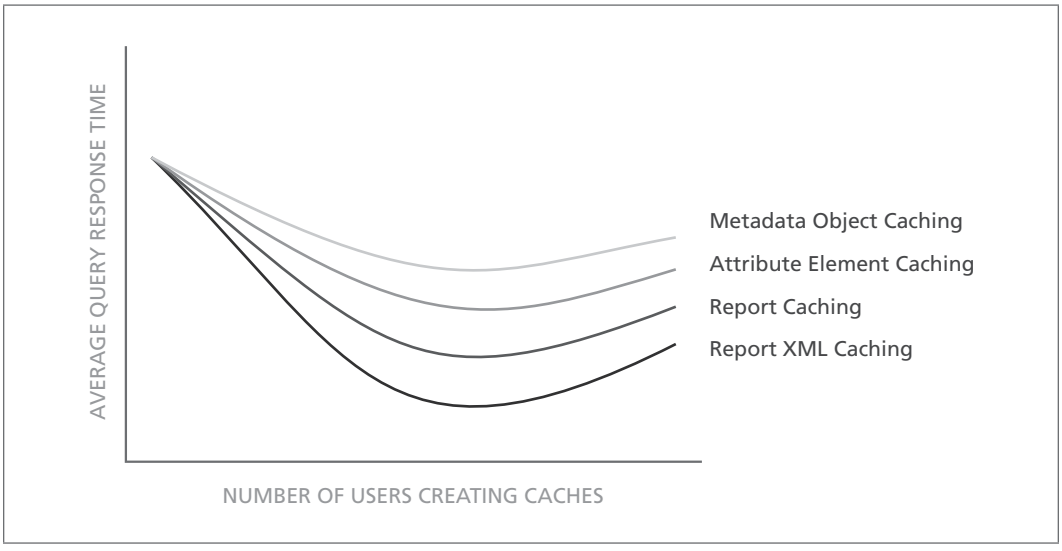


Figure 10.12 Query response time decreases as the number of users increases.

In a clustered Intelligence Server configuration, caches are synchronized across clustered nodes without the massive data transfer between nodes, and users connected to one Intelligence Server node benefit from the caches stored by another Intelligence Server node.

Metadata Object Caching

Metadata objects store information about BI applications that define the data abstraction from the data sources, the business rules and key performance indicators, and the format and layout of documents, reports, and analyses. Users interact directly with these objects and navigate between objects. A responsive system during this interaction is essential for high productivity and unhindered user experience.

Each time a Metadata Object is first accessed, its business definition is cached in memory. When the memory allocation for this cache is reached, the least frequently used object is removed from the cache. Over time, user interaction self-tunes the system to ensure that the most frequently used objects are cached.

Attribute Element Caching

Attribute elements are the row level data for each attribute normally stored in lookup tables in a data warehouse. Lists of attribute elements are displayed when answering prompts, building filters, and browsing the data warehouse. Caching these attribute element lists improves the business user experience and overall system performance.

Intelligence Server optimizes performance by caching the most frequently listed attribute elements and swapping out the least recently used lists. Attribute element lists often span thousands of elements. Intelligence Server employs further techniques for caching this data, and providing rapid response times. As specific sections of attribute element lists are requested, only these snippets of the entire list are initially cached. While the user is perusing the cached section of the list, Intelligence Server prefetches upcoming sections and caches them.

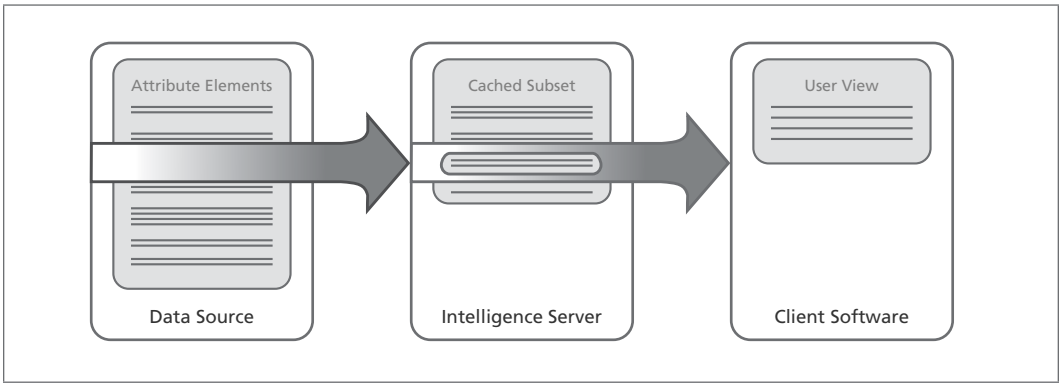


Figure 10.13 Optimized attribute element caching reducing resource requirements while boosting system response times.

Database Connection Caching

When establishing a connection, a database must allocate communication and memory resources, as well as authenticate the user and set up the corresponding security context. To minimize this overhead, the MicroStrategy BI platform implements its own connection pooling strategy via Database Connection Caching. Whenever a connection is established, this connection is kept and re-used when possible. This implementation is database and operating system platform independent, improving the end-user experience and the performance of the BI system.

Report Caching

The MicroStrategy BI platform creates report caches to avoid re-executing requests for exactly the same information against the data warehouse each time, speeding the delivery of information to business users. There are 3 types of report caches in the MicroStrategy BI platform.

1. Result set caches. These store the final results of a report or dataset after Intelligence Server has performed additional analytical processing but before formatting is applied to the results. Result set caches are stored in a highly efficient binary form both in memory and on disk.
2. Formatted PDF and Excel files. Pixel-perfect documents are cached to disk after Intelligence Server has applied all the formatting. Links to these documents appear as messages in individual users' history lists.
3. XML caches. Web receives reports from Intelligence Server in chunks of XML according to the number of rows defined in the "Incremental Fetch" setting, and the contents of the "Page-by" section of a report. These are cached in XML format improving performance for Web users.

Intelligent Cubes Provide High Performance Analysis

As Intelligence Server fulfills report requests from the data warehouse, MicroStrategy OLAP Services creates Intelligent Cubes that model the relational database. Intelligent Cubes give users sophisticated OLAP functionality while maintaining full user access to the entire enterprise data warehouse. The data in the cubes is always up-to-date; if a cube is missing data that the user needs, or if the data has been updated in the data warehouse, the Intelligence Server will retrieve it from the warehouse. Intelligent Cubes present highly interrelated subsets of data to end-users, making it simpler to perform quick analyses. Data proximity accelerates system performance while maintaining user self-service access to all corners of the enterprise data warehouse.

Automatic Data Management

Administrators of other BI software spend significant amounts of time maintaining cubes based on business requirements. The cubes reside in multiple locations, such as user desktops, Web servers, and file servers. The data is duplicated across these locations, and time spent managing the data quickly grows exponentially. In contrast, MicroStrategy Intelligence Server automatically builds and refreshes data. Maintenance is orders of magnitude easier and security is always enforced.

Full Access to Entire Data Warehouse

During regular use, the Intelligence Server automatically accesses Intelligent Cubes whenever relevant without any user intervention. As users analyze the data and expand the scope of their analysis requiring more data, the Intelligence Server automatically accesses the data source on behalf of users. Accessing data from an Intelligent Cube or reaching through to the data source is seamless to users. This provides a dramatic shift in user analysis. Without bounds to a limited aggregate of data, users on the MicroStrategy BI platform are free to navigate the entire data warehouse for comprehensive analysis. IT departments no longer have to build different cubes to serve different business requirements. Instead, most frequently requested data is bundled once, and users self serve ad hoc requests.

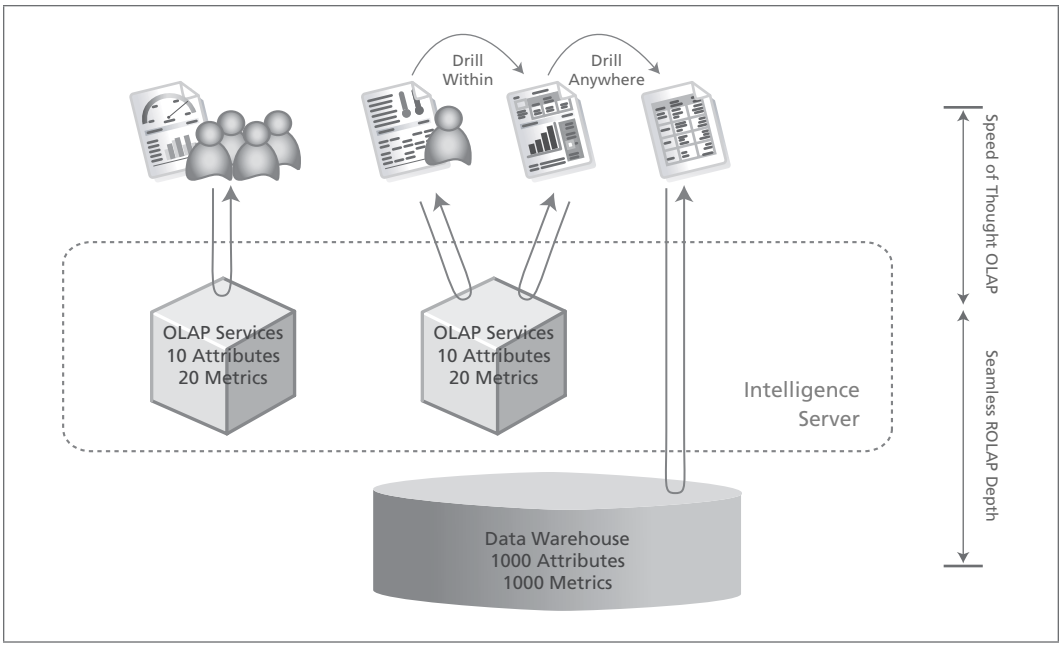


Figure 10.14 Speed-of-thought analysis powered by Intelligent Cubes.

Custom Caching for Maximum Efficiency

To further boost performance, administrator can selectively elect to pre-cache groups of frequently used long running reports for the highest impact while allowing Intelligence Server to create caches on-the-fly, as required, for other reports. Alternatively, caching can be disabled for reports querying against Online Transactional Processing (OLTP) data sources. Intelligence Server statistics provide guidance on what should be cached, with information on most frequently used reports, reports with the largest final result sizes, and reports with the longest query execution times. Armed with this information, the administrator can make intelligent custom caching decisions.

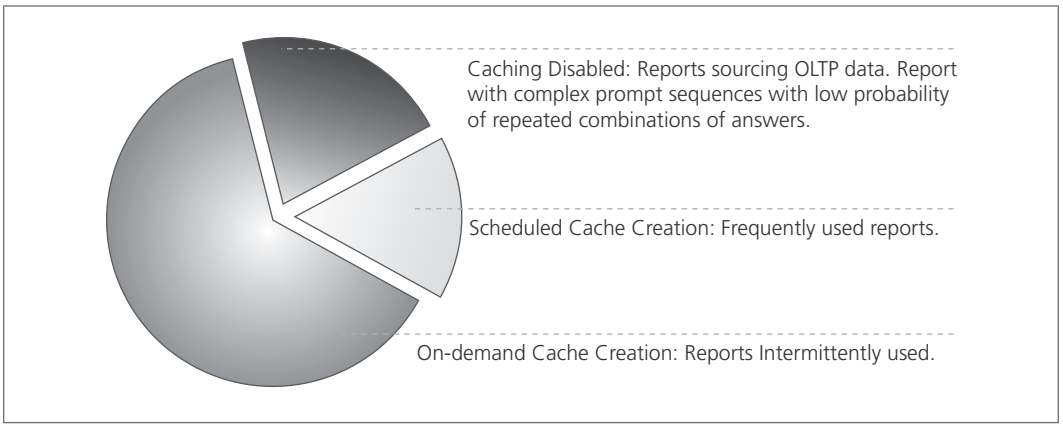


Figure 10.15 Multiple caching strategies applied to all reports ensure efficient caching.

10.3 Distributable Execution

As organizations move toward BI consolidation together with increased data warehouse sizes and more enterprise users, an enterprise BI system needs to be able to handle the increased load. In addition, organization with Service Level Agreements (SLA) requiring predictable levels of service need confidence in their BI system as it grows. These needs are met using distributable execution. MicroStrategy Web, Narrowcast Server, and Intelligence Server are all cluster capable.

Multiple MicroStrategy Intelligence Servers configured in a cluster work cohesively as a single logical unit, and each node services end-users seamlessly. Workload can be distributed across nodes in a cluster, and adding new nodes is straightforward. Resources available to different projects can be dynamically adjusted based on real-time workload. MicroStrategy Intelligence Server clustering enables this via:

- Load Balancing
- Asymmetric Clustering

In addition, clustering allows enterprises to maximize their hardware utilization by starting with a small server implementation, and adding hardware incrementally as the number of end-users increases. As a result, the initial investment in hardware is always preserved, and performance is always excellent.

Load Balancing

Load Balancing refers to a process or algorithm that shares the load on a system among the available servers. Within the MicroStrategy BI platform, load balancing is performed using algorithms on MicroStrategy Web or MicroStrategy Narrowcast Server that route new requests for user sessions to Intelligence Server with the most available open sessions.

Once a session is established, all requests from a connected user are directed to the same Intelligence Server. Load balancing ensures that response time is maintained as the number of end-users increases. The additional users are distributed across multiple servers which process jobs simultaneously.

Unlike other BI products where the load balancing algorithm cannot be customized, load balancing within an Intelligence Server can be customized by the load balance factor. With MicroStrategy Web and MicroStrategy Narrowcast Server, each node within an Intelligence Server cluster is assigned a work load weight; user connections are distributed to reflect the ratio of these weights. As a result, business needs or hardware considerations such as the following can be incorporated into the MicroStrategy BI platform:

- Upgrade Transition
- Mixed Clients Deployment
- Geographic Proximity

During an upgrade cycle, Intelligence Server nodes are upgraded in a rolling window to minimize risk and avoid downtime. After a server is upgraded, the load balancing factor can be used to give the upgraded machine a heavier load.

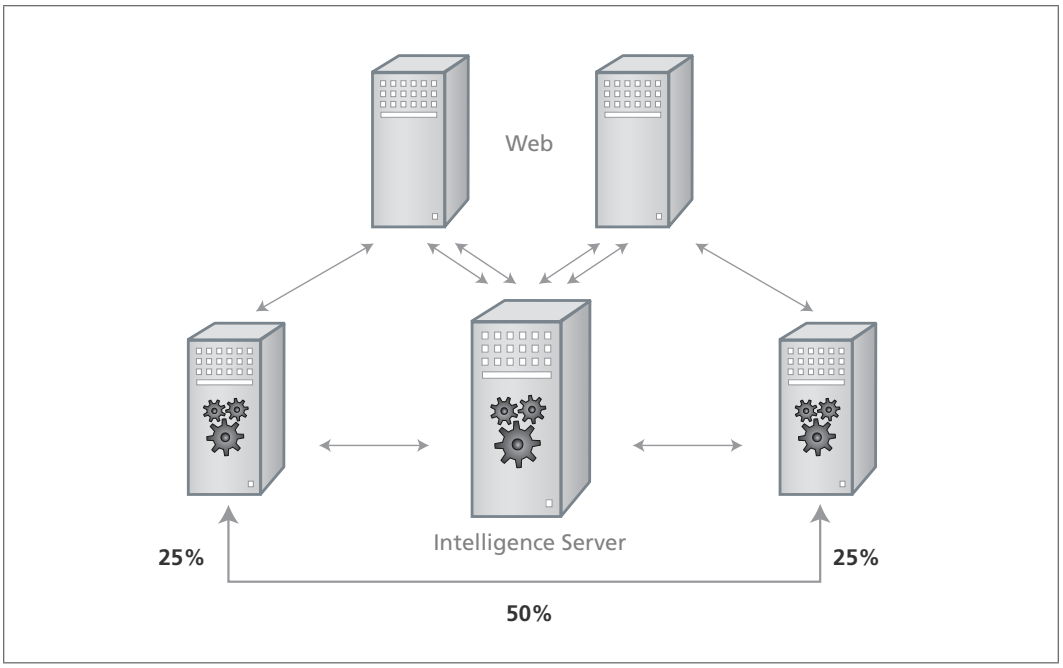


Figure 10.16 Load balancing across Intelligence Server nodes is tunable depending on each node's hardware resources.

An organization with a mix of Desktop power users and Web end users can decide to dedicate an Intelligence Server to Desktop users due to their usage profiles. If there is a small number of Desktop users, resources available on the dedicated node may not be used heavily, but can peak sporadically. The organization can choose to divert a small amount of Web user load onto this server to capitalize on the idle resource, and speed up end-user experience for the Web users.

An organization with a multi-location environment can choose to point each location to a particular Web server, and configure the load balancing factor of that MicroStrategy Web to favor the closest Intelligence Server node.

Asymmetric Clustering

MicroStrategy Intelligence Server provides Asymmetric Clustering, in which each Intelligence Server node within a cluster can host a different set of projects. This feature enables further opportunities to optimize the overall BI application, better utilizing available resources, and increasing performance. Asymmetric clustering in MicroStrategy Intelligence server allows:

- Further optimized resource allocation
- Increased agility to react to usage fluctuations
- Enhanced Failover and Fallback Logic

Additional load balancing configurations are available for MicroStrategy Web, MicroStrategy Office, and Narrowcast Server.

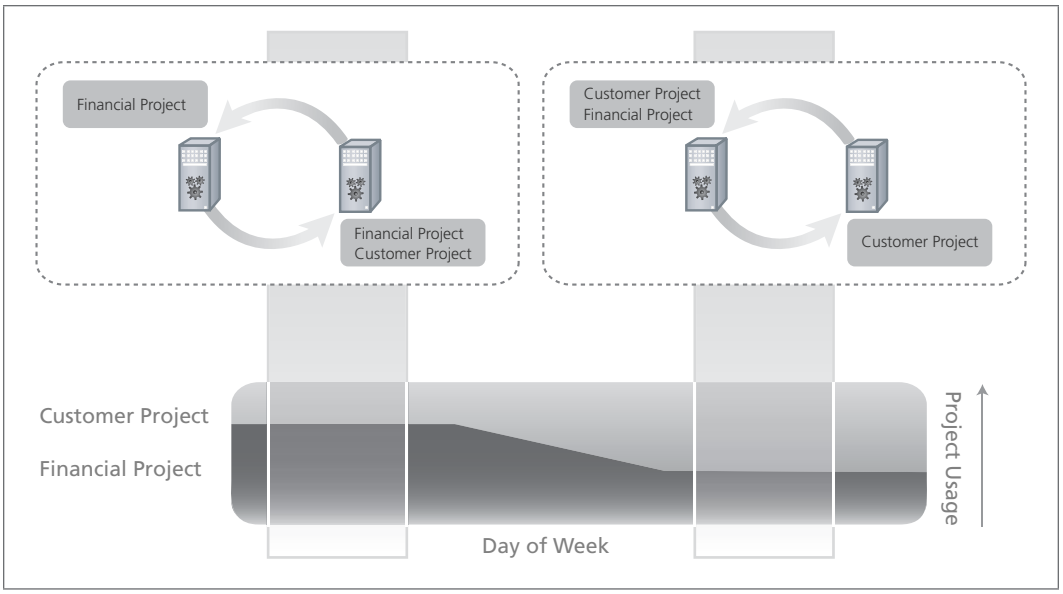


Figure 10.17 Asymmetric clustering in Intelligence Server spreads the load of the BI applications across nodes.

10.4 64-bit Business Intelligence

Continuing to spearhead business intelligence system scalability and performance, MicroStrategy began development work towards true 64-bit technology in 2001. Part of this challenge was to incorporate a new technology framework compatible with all operating systems on any architecture.

Envisioning the boost in scalability and performance, MicroStrategy started incorporating parts of this technology framework in MicroStrategy offerings in 2002. Development in this direction continued until its completion, and the release of MicroStrategy Universal in 2004. With a single code base, there is functional equivalence across all platforms. All platforms can be released simultaneously, and businesses choosing to leap to 64-bit MicroStrategy Business Intelligence technology can simply switch out the software supporting their existing implementations.

Leaping from 32-bit technology to 64-bit technology provides a gargantuan capacity increase for a business intelligence system. The more memory the business intelligence system can track, the larger the data sets it can manage and hold, and the faster it can service users.

Most business intelligence vendors only offer 32-bit solutions where the addressable memory is limited to 4 gigabytes. Others may execute on a 64-bit environment but may not be optimized for a 64-bit environment, and still be limited to 4 gigabytes of addressable memory. A true 64-bit solution can address up to 16 exabytes of memory. As a result of being a true 64-bit technology, MicroStrategy BI systems can support:

- Increased number of users
- Increased performance through increased cache capacity
- Richer BI applications through increased metadata capacity
- Reduced numbers of servers

True 64-bit MicroStrategy BI technology seamlessly scales with the growth in BI applications, users, and data sizes while delivering ultra high performance.

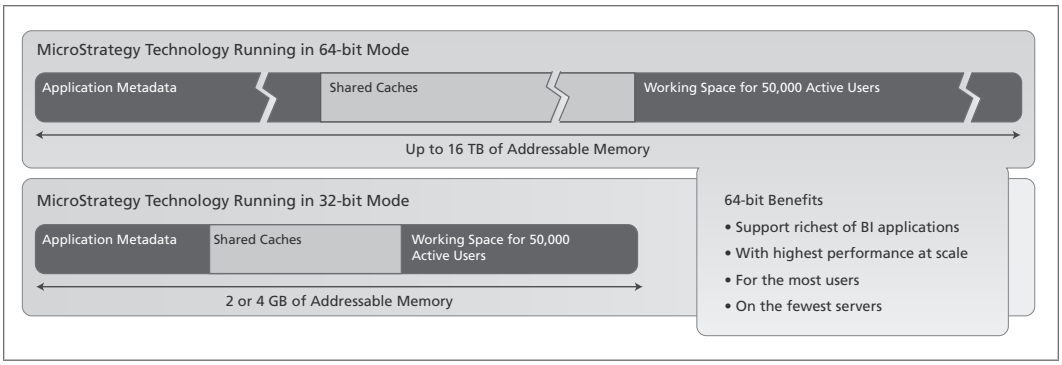


Figure 10.18 64-bit MicroStrategy technology supercharges BI allowing more users, applications, and caches.

More Users for Enterprise Wide Reporting & Analysis

With MicroStrategy 64-bit technology, far more users are able to tap into the largest databases, and obtain answers to their reporting and analysis inquiries. Information collected throughout business operations becomes available to the entire enterprise instead of select groups of individuals. Critical and active analysis is now available throughout the organization.

More Sophisticated and Comprehensive Analysis with Complete Data Warehouse Coverage

With MicroStrategy 64-bit technology, more capacity provides the opportunity to perform multi-layered, multi-level complex analysis on the entire data warehouse beyond the bounds of the database by business users. Analysis is no longer bound by 32-bit limits. BI systems have more capacity to match the growth of enterprise data warehouses, and to cover all aspects of information gathered and stored inside. More sophisticated questions can be answered to shed more comprehensive insight into the entire business operation.

Reduced Number of Servers Drives Enterprise Standardization

With increased capacity comes the opportunity to consolidate. A single BI system can support more users with better performance and richer applications. The management of BI systems is simplified with a reduced number of servers. MicroStrategy 64-bit technology provides a new level in support of an organization's BI standardization efforts.

10.5 Multi-Threaded Processing

Compared to other BI products, MicroStrategy Intelligence Server is a fully multi-threaded application that consists of a group of well-defined components. Each component performs specific tasks, and each task is performed by multiple processing threads. This approach, called multi-threaded processing, leverages all available processors in a system, giving it vertical scalability. According to Gartner Research:

"This technique enables an application to support many users simultaneously with a single image of the software. Instead of executing a separate copy of the application for each user requesting services, a single instance of the application spawns "threads" or sub-processes, as needed, to support concurrent users. This imposes less of a load on the server, and does not require the initial startup time for each new instance of the application. Additionally, threads can take advantage of multiple processors, if they are available, further enhancing scalability. Developing multi-threaded code is difficult, especially in the debugging stage. Consequently, many BI software providers do not support multithreading, or they offer only limited support."

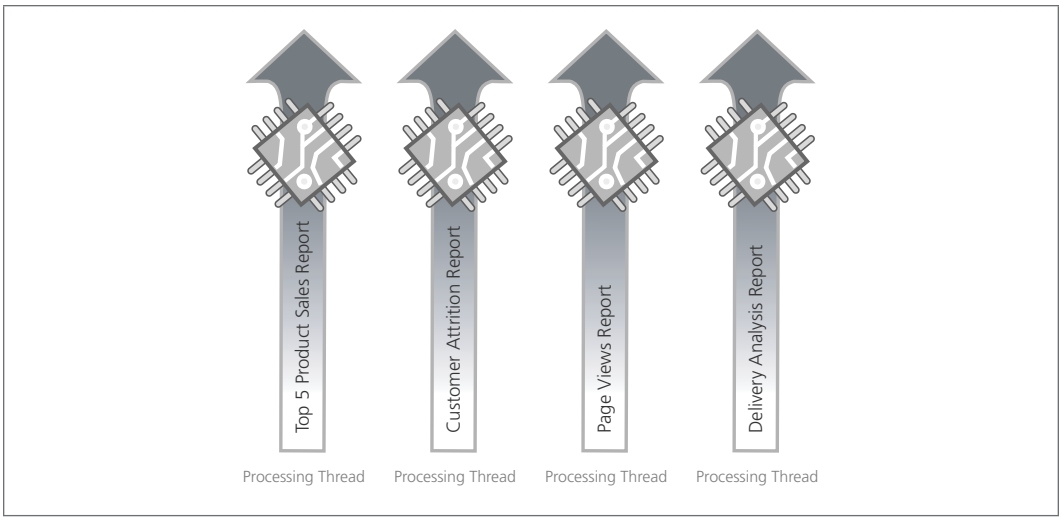


Figure 10.19 Multi-threaded processing concurrently executing reports on multiple processors.

The following are examples of multi-threaded components within MicroStrategy Intelligence Server. Each component is assigned a certain number of threads. The higher the number of threads allocated to a component, the more capacity it has.

- Object Browsing Engine
- Resolution Engine
- Command Engine
- Dynamic SQL Engine
- Dynamic MDX Engine
- Query Engine
- Formatting Engine
- Analytical Engine

The demand for individual components will vary depending on the numbers and types of job requests for report creation, report execution, and data manipulation being processed. For example, when many users request different queries simultaneously, the Dynamic SQL Engine, being one common component, experiences a load increase. When the report results have been retrieved from the data source, processing shifts to the analytical and formatting engines.

To prevent a bottleneck in any one of these components, Intelligence Server employs an advanced automatic resource balancing scheme. This consists of dynamically allocating resources for each component based on its immediate needs. In the above scenario, the Intelligence Server will initially allocate more threads to the Dynamic SQL Engine. When the Analytical and Formatting Engines take over, Intelligence Server dynamically increases the number of threads for these components while reducing the number of threads for the Dynamic SQL Engine. This adaptive balancing of internal capacity ensures effective use of available resources, and allows the MicroStrategy BI platform to scale while maintaining excellent performance.

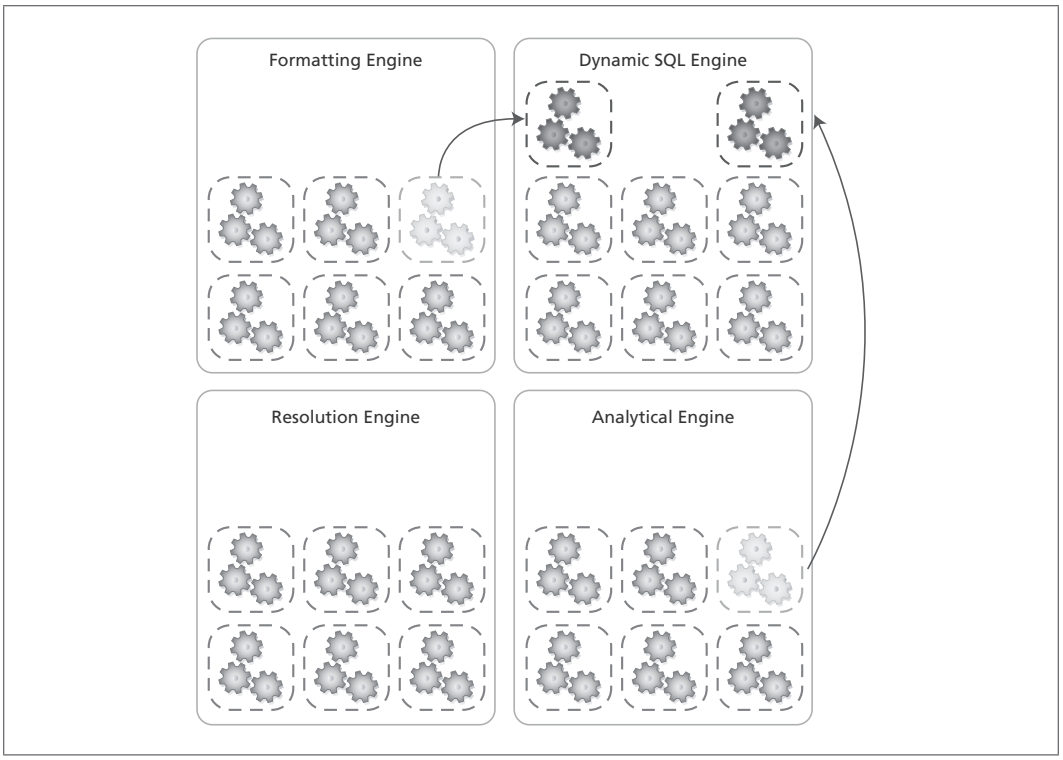


Figure 10.20 Distributing processing threads between processing engines automatically removes performance bottleneck.

10.6 Efficient Communications

The MicroStrategy BI architecture is designed to minimize network traffic between the different components within the architecture. While each component handles different parts of the user query, coordination and communication between these components are required.

Network traffic from the data sources and metadata repository to the Intelligence Server, coordination between nodes of an Intelligence Server cluster, and communications between client interfaces and the Intelligence Server are network efficient.

Interacting with Data Sources

When Intelligence Server interacts with data sources, only the required data is brought back, and as much aggregation as possible is performed at the data source. When data needs to be written back to an RDBMS, parameterized queries are used to reduce the net amount of data transferred. When querying a large attribute element lookup table, only the rows that are needed are retrieved and cached. When accessing business definitions within the Metadata Repository, Intelligence Server caches the most frequently requested definitions to reduce traffic.

Report Snapshot Delivery

With other BI products, when reports containing thousands of rows are returned to the client, the transmission process can slow down or be limited by the volume of network traffic. Even as bandwidth becomes cheaper, optimizing data transmissions remains important as the data still requires processing for client rendering. Just as search engines results typically display 20 hits per page, despite finding 30,000 hits or more, the MicroStrategy BI platform returns results to the end user in 20-row snapshots. This number is configurable by the end user by adjusting the incremental fetch. Users have the ability to click "Next," or select a page number, to view the next batch of results.

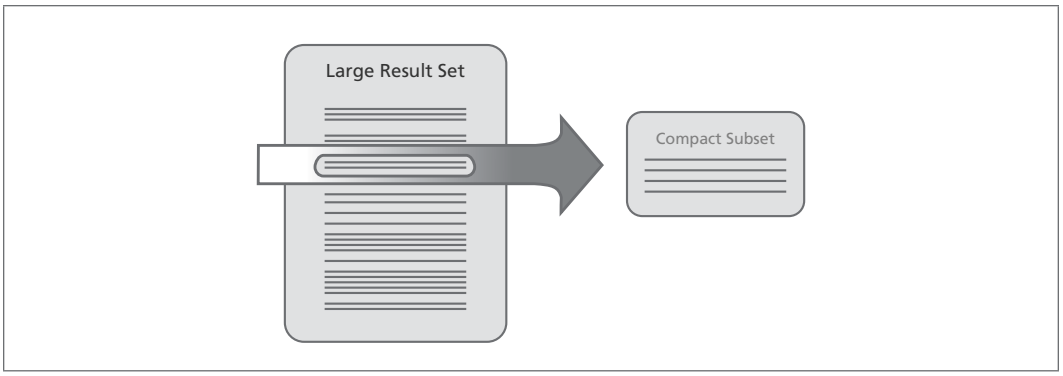


Figure 10.21 Report snapshot delivery incrementally fetches needed subsets of results to users.

Report snapshot delivery minimizes network traffic while maximizing performance as the amount of data processing on client machines is minimal.

Efficient Report Manipulation with Drilling

Drilling anywhere allows users to simultaneously navigate through the entire data warehouse and perform train-of-thought analysis. Other BI products require users to retrieve all items, all regions, and all months for drilling. Instead of this approach, which increases network traffic and poses data volume limitations, the MicroStrategy BI platform drill anywhere feature allows users to specify the slice of data for drilling. For example, when a user wants to drill from a specific item to geographic information, only the geographic information for that particular item is retrieved. In addition, drill paths can be customized per report, giving guidance to end-users and reducing the number of errant jobs.

Personalized Page-by Execution

MicroStrategy Narrowcast Server's personalized page-by execution mode allows a single report to be parsed into sub-reports for individual subscribers. Requests are consolidated, saving on overhead, and later sliced into specific subsets for report bursting to users. Consequently, Narrowcast Server can deliver thousands of personalized services while ensuring a minimal load on the data warehouse. Personalized page-by execution is especially useful for personalized reports with common themes. Rather than running 1,000 different reports for 1,000 different users, a large report can be run just once. Selected rows of the large report are sent to recipients based on their personalization settings.

With personalized page-by execution, the processing workload is distributed across architectural tiers, and communication overhead between the Intelligence Server and data source is minimized. In addition, overhead within the data source is also minimized as only one query is requested.

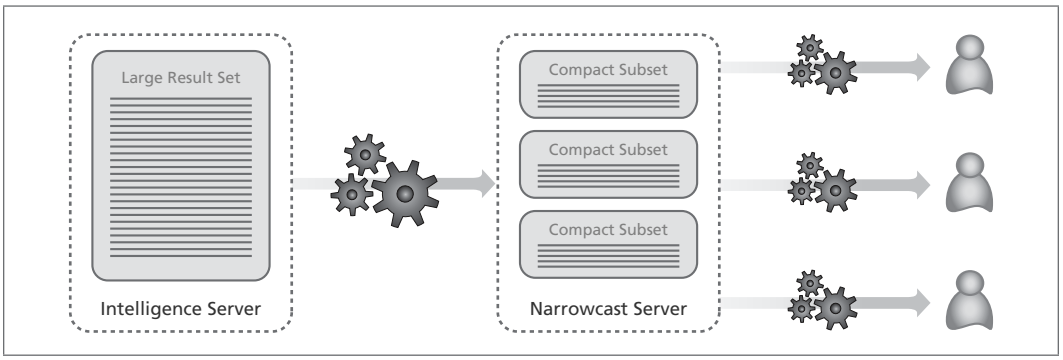


Figure 10.22 Personalized page-by distributes individual reports from a single Intelligence Server query.

10.7 Proven Scalability and Performance

With all of the above technologies, the MicroStrategy BI platform leads the pack in both scalability and performance. By embedding the goals of building an architecture with world class scalability and performance into the design process, MicroStrategy sets the standard for these characteristics. With each user query, multiple complementary technologies work hand-in-hand to boost performance, and reduce total response time. By servicing users in the most expeditious manner, resources remain available for other users.

1,000 – 5,000 Users				
Telefónica	Telecom	<250 GB	Oracle, SQL Server	2,000 Users
Carl Warren & Co.	Insurance	<250 GB	SQL Server	>2,000 Users
Amway	Consumer Goods	1.5 TB	SQL Server 2000	>2,000 Users
U.S. Dept. of Education	Government	500 GB	Oracle	4,000 Users

5,000 – 20,000 Users				
Hudson's Bay Company	Retail	2 – 3 TB	Oracle, Teradata	5,000 Users
Ace Hardware Corp.	Retail	>1 TB	Teradata	8,600 Users
ARC	Travel	25.5 TB	Teradata	15,000 Users
Sprint	Telecom	>1 TB	Teradata, Oracle, Informix	16,000 Users
Bavarian State Ministry of Sciences, Research & the Arts	Education	250 GB – 1 TB	SQL Server	17,000 Users

>20,000 Users				
US Postal Service	Government	33 TB	Teradata	>20,000 Users
British Telecom/ Xansa	Telecomm.	4 TB	Oracle	30,000 Users
European Medicines Agency	Pharmaceu-tical	In Dev.	Oracle	Up to 40,000 Users
Northwest Evaluation Association	Education	In Dev.	SQL Server	Up to 100,000 Users

*Data collected as of May 2005. User bases are expected to expand further.

<3 Terabytes				
Charming Shoppes, Inc.	Retail	>2 TB	Teradata	800 Users
Dick's Sporting Goods	Retail	2 TB	Oracle	250 Users
Bank of Montreal	Financial Services	2 TB	IBM DB2	2,000 Users
H-E-B	Retail	2.5 TB	Teradata	800 Users

3 – 10 Terabytes				
Cox Business Service	Telecomm.	3 TB	Oracle	100 Users
LoanPerformance	Financial Services	3.5 TB	SQL Server	90 Users
Lotte Mart	Retail	3.6 TB	Teradata	500 Users
Grange Insurance	Insurance	4 TB	SQL Server	300 Users
Autotrader.com	Retail	7 TB	Oracle	800 Users

>10 Terabytes				
eBay	Retail	12 TB	Teradata	300 Users
Ahold	Retail	15 TB	Oracle	1,000 Users
Lowe's Companies	Retail	16 TB	Teradata	>2,000 Users
METRO Group	Retail	34 TB	Teradata	2,500 Users

*Data collected as of May 2005. Data sizes are expected to increase further.

Figure 10.23 Customer proven scalability and performance with the MicroStrategy BI platform.

Large numbers of customers using the MicroStrategy BI platform are able to scale their BI deployments using the MicroStrategy BI platform both in terms of the number of users, and the amount of data accessible to them. Proven scalability and performance are hallmarks of the MicroStrategy BI platform.

11

Exporting Data and Reports

Most MicroStrategy users work online, running live reports against the most current information loaded in the data warehouse. MicroStrategy adds analytical context to this data, making it much more valuable for monitoring business processes, enriching enterprise reports, and improving analytical decision making.

11.1 MicroStrategy Exports Enhanced Data

Enhanced information is often needed outside MicroStrategy's interfaces and applications for:

- Offline Analysis – obtaining data while connected to a network, and then disconnecting from that network before analysis is performed
- Application Integration – sharing data with other applications, for example ERP, Supply Chain Management, and Budgeting systems, without human intervention
- Offline Review – generating files in a static format, such as HTML, PDF, and Microsoft Word and PowerPoint
- Batch Printing – distributing reports directly to any network printer in an industry-standard printable format
- Report Archives – storing reports and documents of any size in specified file locations for recording purposes

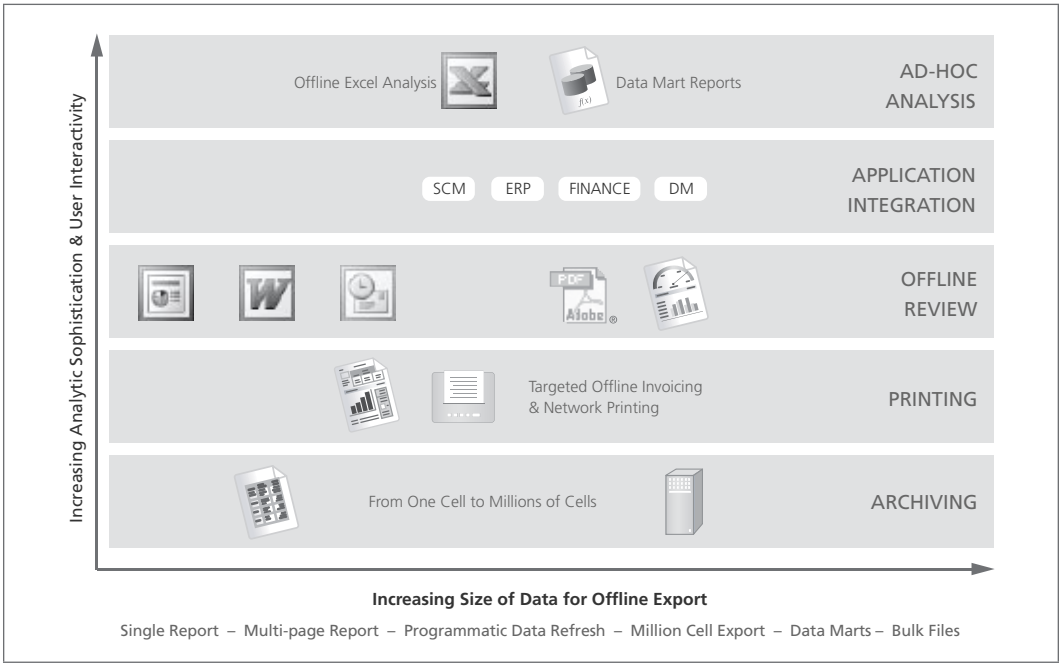


Figure 11.1 Offline use cases arranged by user interactivity, data size, and frequency of export.

11.2 MicroStrategy Provides Efficient Exporting

Requests for data required outside the MicroStrategy BI platform are initiated differently, depending on how and where the data is going to be used. MicroStrategy provides four main approaches to fulfilling such requests:

- Exporting data directly from the user interfaces
- Saving reports in datamart tables in a database
- Performing bulk exports with Narrowcast Server
- Integrating with open APIs and Web services

Each of these methods addresses the requirement that the right data is transferred quickly and efficiently in a useful format. Deciding which method to use depends on the following factors:

- Data Accuracy
- Data Volume
- Data Format

Data Accuracy Keeps Export Processing to a Minimum

MicroStrategy contains many ways to control the accuracy of the data in a report. Ensuring that only the necessary data is exported reduces the amount of processing required to package and consume report information. Besides the standard filtering contained in reports and documents, this is accomplished via user input, ad hoc navigation, prompted reporting, and an automated security process.

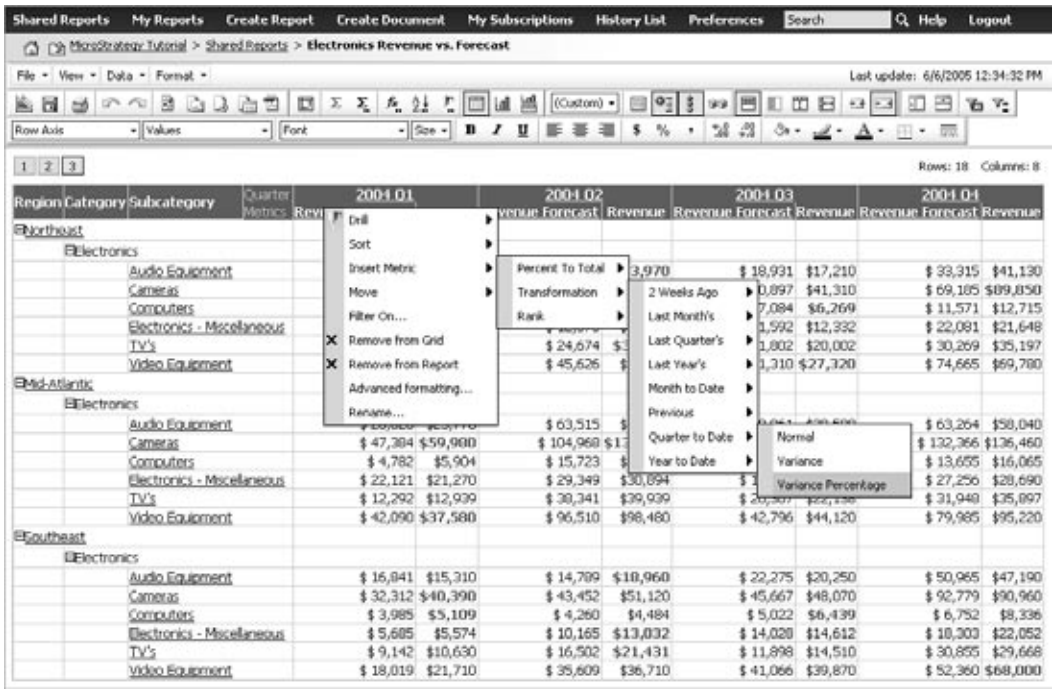


Figure 11.2 Users navigate to the data they need with a right-click menu.

Ad hoc navigation is performed naturally by a user, who changes the initial context of a report using standard analytical methods such as pivoting, page-by, sorting, and drilling. When this navigation results in a useful set of data for offline analysis, the user can export the data immediately if the result set is relatively small. For large result sets, the information can be saved as a datamart table or a standard MicroStrategy report that the Narrowcast Server can process.

Prompted reports efficiently personalize the data retrieved through simple questions and answers. More accurate data is initially displayed, reducing the amount of ad hoc navigation needed before exporting the data. If the report is saved before exporting, the user's actual prompt answers are remembered for future use.

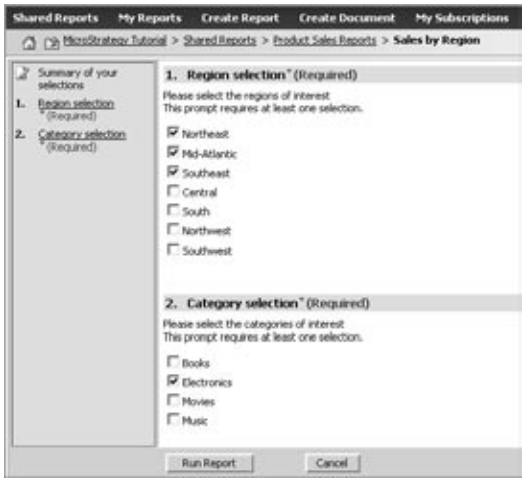


Figure 11.3 A MicroStrategy prompt in a standard report.

User permissions add another tier to ensure data accuracy. Security filters and object access control lists are automatically applied to any report requested by a user. Even reports that are run by Narrowcast Server on behalf of a user are subject to this security layer.

Data Volume Exported Varies by Use

The data volume exported from the MicroStrategy BI platform ranges from a single row or cell of data to many millions of rows. Small, manageable datasets are usually exported directly from user interfaces. These are typically composed of a few screens or pages, and can span up to several thousand rows of data. Export actions are embedded in all MicroStrategy interfaces. MicroStrategy Office is designed to handle larger datasets than the other user interfaces.

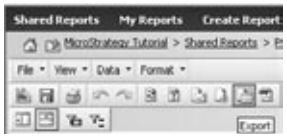


Figure 11.4 The Export buttons in MicroStrategy Web.

Certain third-party applications, such as data mining tools, require datasets that are several gigabytes in size. These datasets should be saved as tables in a database using dynamic datamarts, or exported using Narrowcast Server's bulk export feature.

The amount of data that must be transferred and the total number of distinct export requests affect the time taken to retrieve the data from the data warehouse, apply additional analytical processing, formatting, and transfer the data. The MicroStrategy BI platform utilizes both Intelligence Server and Narrowcast Server for efficient processing of export requests. Intelligence Server manages immediate exporting that is optimized for delivering small datasets quickly, while Narrowcast Server is designed for bulk information delivery.

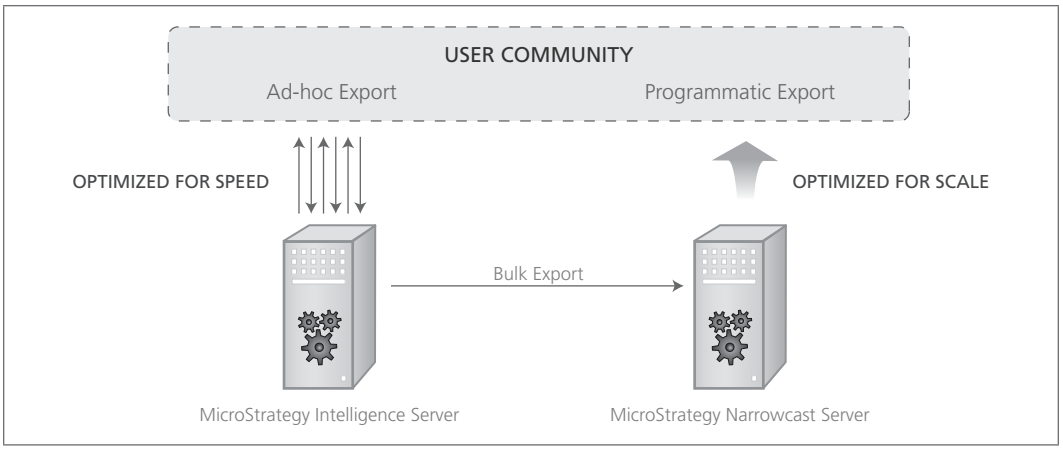


Figure 11.5 Exporting in MicroStrategy is optimized for different use cases.

Report Data is Exported in Many Output Formats

Depending on the use of the data, the required output format varies significantly. Ad hoc offline analysis is typically done in Microsoft Excel. Offline review is normally conducted in Microsoft PowerPoint, Microsoft Word, HTML, and PDF formats. In most of these cases, the precise formatting of the MicroStrategy report must be kept in the final output. Indeed, MicroStrategy applies the formatting stored in the metadata repository to exported data when required.

When larger amounts of data are exported, detailed formatting of the data itself is generally not necessary. The output format is of more importance, as large datasets are mostly imported into other applications. Formats used for these datasets include database tables, text, and Web services formats.

Narrowcast Server delivers data and reports in emails as HTML or text content, and as PDF, Excel, and text file attachments. The attachments can be compressed automatically before being sent. Narrowcast Server also delivers reports to printers, and to file systems for offline review and report archiving.

MicroStrategy has invested significant resources into supporting a wide range of output formats for thousands of customers worldwide who use many different RDBMS, and hundreds of thousands of report designs. Along the way, MicroStrategy has addressed significant challenges, including:

- Repeating row and column headers over multiple pages
- Incorporating correct HTML codes
- Optimizing inserts for each supported database
- Handling data formatting issues such as:
  - null values
  - leading zeros
  - negative values
  - spaces
  - double-byte characters

MicroStrategy Office formats data using information in the metadata. This formatting can be changed in the Microsoft documents directly, and will persist if the data is refreshed. MicroStrategy Office supports the widest range of Excel formats:

- Grids
- Charts
- Outlines
- PivotTables®
- PivotCharts®
- Flattened
- Quickgrids



Figure 11.6 MicroStrategy Office gives users many options for running reports with different output formats in Microsoft Office.

Different components in the MicroStrategy BI platform convert report data into the desired data format. For some formats, such as PDF, the formatting can be distributed across more than one component in order to spread the processing across more than one server. Exporting to the following formats is available from MicroStrategy:

Export Format	Formatting Component	User Interface
HTML	MicroStrategy Web	MicroStrategy Web, Desktop
PDF	MicroStrategy Intelligence Server, Narrowcast Server	MicroStrategy Web, Desktop
Excel	Intelligence Server, Narrowcast Server, MicroStrategy Web	MicroStrategy Web, MicroStrategy Office, Desktop
PowerPoint	Intelligence Server, MicroStrategy Web	MicroStrategy Office
Word	Intelligence Server, MicroStrategy Web	MicroStrategy Office, Desktop
Access	Intelligence Server, Desktop	Desktop
Email	MicroStrategy Narrowcast Server	N/A
Zip	MicroStrategy Narrowcast Server	N/A
Web Services	MicroStrategy Web	MicroStrategy Office
Data Mart	MicroStrategy Intelligence Server	N/A
CSV Text Files	MicroStrategy Web, Intelligence Server, or Narrowcast Server	N/A

Figure 11.7 MicroStrategy's user interfaces export report data to many different formats.

11.3 MicroStrategy Office Integrates With Excel, Word, and PowerPoint

Microsoft Office is the most prevalent desktop productivity suite. MicroStrategy Office runs MicroStrategy reports for users of Excel, Word, and PowerPoint, extending business intelligence to many more people. Simply put, MicroStrategy Office allows Microsoft to be the interface to the business intelligence system with all of the security, robustness, and data consistency of the MicroStrategy BI platform.

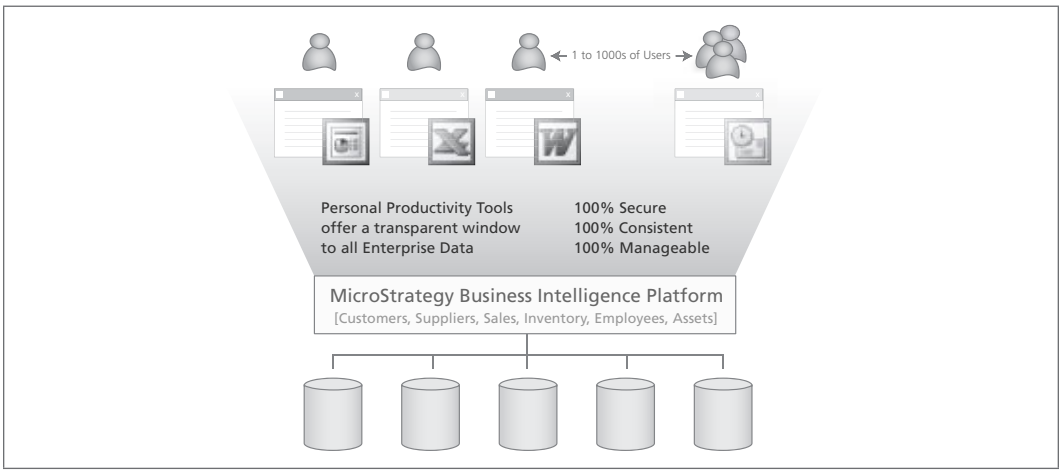


Figure 11.8 MicroStrategy Office runs reports directly in Microsoft Office, extending business intelligence to many more people.

Integration with the productivity suite is robust, because MicroStrategy Office uses shared Microsoft Office components, adheres to the Microsoft .NET Architecture for Office, and is coupled to the MicroStrategy BI platform using Web services. Data is transmitted in SOAP packets in XML or CSV formats. The lightweight .NET add-in provides native support for Microsoft Office worksheets, charts, PivotTables, and PivotCharts. The same formatting viewed in MicroStrategy's standard interfaces is applied to reports and documents run from Microsoft Office. Charts can be presented as live Microsoft Office charts or as images from Intelligence Server. Additional formatting added in the Microsoft Office document is automatically retained even if the data is refreshed.

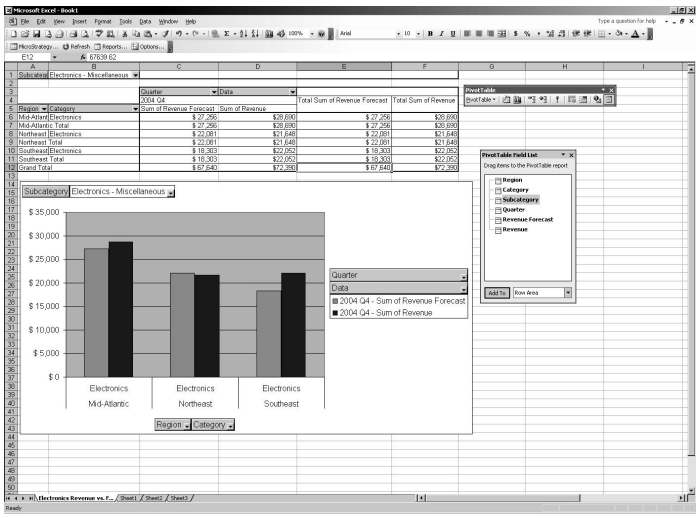


Figure 11.9 A MicroStrategy report can be run simultaneously as a PivotTable and PivotChart. Using the PivotCache, millions of rows can be analyzed offline.

MicroStrategy Office is particularly efficient for exporting workbooks that contain many reports. Persistence data is stored in the Microsoft Office document, and an entire document, potentially containing dozens of MicroStrategy reports, can be completely refreshed with new data with just one click.

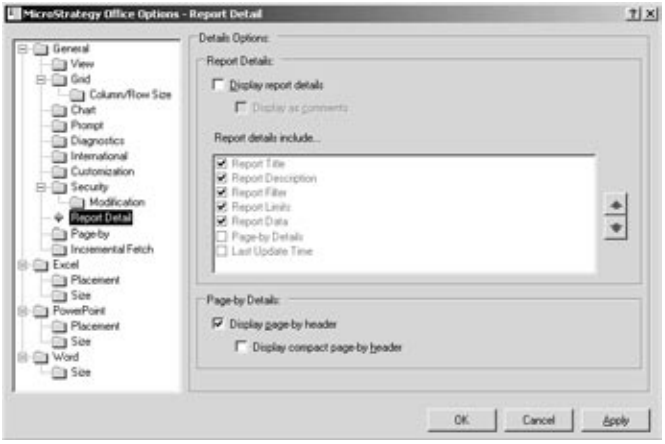


Figure 11.10 MicroStrategy Office options provide the flexibility that the Microsoft Office developer requires.

Beyond the simple exporting of reports, the Microsoft Office integration takes advantage of browsing and searching MicroStrategy reports, accessing the History List, and re-prompting. The user can also access many projects in series. All enterprise security enforced by MicroStrategy Intelligence Server is also enforced for Microsoft Office users. For developers interested in extending these capabilities, MicroStrategy Office has a complete programming API.



Figure 11.11 MicroStrategy Office persistence data stored in the Microsoft Office document.

11.4 MicroStrategy Supports Flexible Exporting for All Uses

The MicroStrategy BI platform offers the broadest, most scalable exporting options for extracting analytical information from data warehouses and other data sources. MicroStrategy Web, Desktop, and MicroStrategy Office users can export most report data volumes directly from their preferred interfaces into an output format of their choice. Narrowcast Server sends report information to email systems, file systems, and printers, and can handle very large files. For the dominant use cases in exporting and offline use, MicroStrategy offers an efficient, robust, and scalable solution.

12

Data Mining and Predictive Analysis

While providing sophisticated historic analysis of an organization's business has been the mainstay of BI applications, using the historical information to generate, and distribute predictive insight is often a challenge for organizations. In search of predictive analysis capabilities, organizations have invested in homegrown utilities and standalone data mining products. These data mining tools incorporate statistical analysis, modeling techniques, machine learning, and database technologies to search for hidden patterns in historical data. Organizations develop predictive models from these hidden patterns, and use them to assess the likelihood of future events.

Despite the practical value of developing predictive models with data mining tools, adoption of predictive analysis by mainstream business users has been slow. Some commonly cited reasons for poor adoption of data mining by the business user community include:

- Data mining practitioners must be highly skilled in statistical analysis, modeling techniques, numerical analysis, database technologies, and process knowledge. These highly specialized skills are not the core competency of typical business users who need to use the predictive insights to make everyday decisions.
- Data mining tools often require the extraction of sample data from the enterprise data store for use with the data mining tools. Data samples are subject to pre- and post-processing so that the data mining tool can use them. The data extraction and subsequent pre- and post-processing often lead to islands of information, differences in business definitions, and the challenge of reconciling information between the BI system and data mining tools.
- Implementing data mining requires practitioners to learn and understand these specialized data mining products. Since the data mining products are separate products, distinct from the BI applications, they represent yet another set of tools that business users need to master.
- Perhaps the most common reason for poor adoption of data mining is the inability to distribute the predictive models in an easy-to-consume, highly formatted, and timely manner that is integrated with other BI reports and analyses. Integration of data mining into the business intelligence platform has to satisfy these requirements:
- Offer various options for generating predictive analysis for everyday reporting.
- Make it unnecessary for business users, who must consume the predictive insight to be highly skilled in the data mining technology and predictive algorithms.
- Provide a consistent, unified interface for the distribution and consumption of the predictive analysis that can be incorporated into scorecards, dashboards, enterprise reports, advanced analysis, and alerting.
- Re-use a corporation's existing investment in data mining products.

12.1 Integrating Data Mining Into Business Intelligence

In general, data mining software assists and automates the process of building and training highly sophisticated data mining models, and applying these models to larger datasets. The data mining process involves the following steps:

1. Creating a predictive model from a data sample
   A sample of data with a known outcome is extracted from the enterprise data store and pre-processed for the development of the predictive model. Advanced statistical and mathematical models are used to identify the significant characteristics and trends using the pre-processed fields as inputs, resulting in a predictive model. Generally, only a small subset of the all characteristics and trends in the sample data is used in the model.

2. Training the model against the dataset and its known results
   The new predictive model is applied to additional data samples with known outcomes to validate whether the model is reasonably successful at predicting the known results. This gives a good indication of the accuracy of the model.

3. Applying the predictive model to a new dataset with an unknown outcome
   Once the predictive model is validated against the known data, it is used for scoring, which is defined as the application of a data mining model to forecast an outcome. For example, a data mining model that predicts the likelihood of a customer responding to a marketing campaign will generate a score for each customer that indicates his or her likelihood to respond. This score can be a simple binary result, such as a "Yes" or "No," or it could be a number indicating the propensity or confidence in that customer responding, say "97%."

As mentioned earlier, the "Create-Train-Apply" process is typically the domain of the statistician or the data mining analyst. A solid understanding of data mining concepts, statistical concepts, techniques, and data mining tools is necessary in the "Create" and "Train" steps. Applying the predictive model requires less expertise, and is available for all business users.

Scoring Data for a BI Application

There are three main approaches to integrating predictive insight into a BI application:

1. Data mining tool scores the database
   The data mining tool scores the records in a batch process, and saves them as columns in database tables. The BI application references the scored columns as required.

2. Database does the scoring
   Database uses embedded scoring algorithms to score records in response to SQL queries from a BI application.

3. BI application does the scoring
   The BI Platform scores records using scoring metrics and reports.

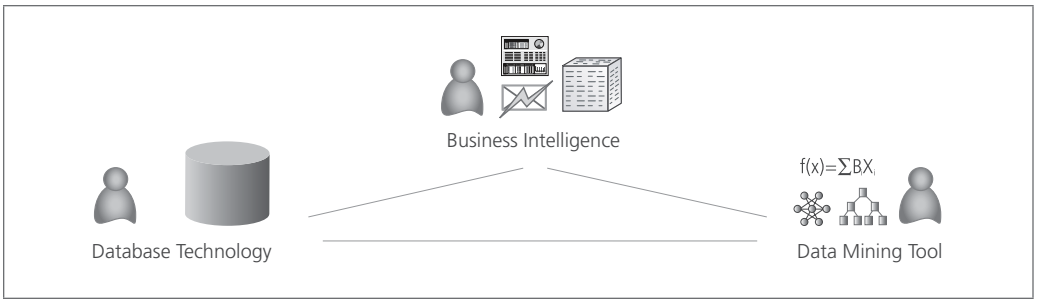


Figure 12.1 Scoring can be performed by any one of the three components of enterprise BI applications, i.e., data mining tool, database, or the BI application.

All three approaches are viable methods for deploying data mining results throughout the enterprise. Determining which approach to use depends greatly on the business need for predictive analysis, and the IT infrastructure and philosophy.

The starting point for most data mining implementations is to use the data mining tool for scoring. Although it is very common for the data mining analyst to provide scores in standalone flat files or spreadsheets, integrating scored results into databases has long been a common practice.

When scoring is required on a real time basis, or when predictive models are created, and changed faster than scores can be calculated and stored in the database, one of the other approaches must be adopted. If the database supports data mining, deploying models in the database is a possible next step. If the BI Platform contains data mining capabilities, deploying models directly in BI applications can speed the adoption of predictive analysis by business users.

Irrespective of which approach is suited for an organization's predictive analysis needs, the MicroStrategy BI platform can integrate predictive insight into all BI applications used by business users. The following sections discuss the benefits and drawbacks of each approach.

Data Mining Tool Does the Scoring

In this approach, a data mining scoring application calculates, and inserts scores into the database as new tables, new columns in existing tables, or updates to existing (old) scores. Once the scores are part of the database, the BI application reads these scores just like any other data, directly from the database. Historically, this approach has been the most common, and has the following benefits and drawbacks:

Benefits:

- Since a data mining tool does the scoring, model complexity, and performance is hidden within the scoring engine. The scoring process does not require any BI resources, and should not impact other concurrent BI processes.
- At runtime, BI applications simply read the scores from the database without having to calculate scores on the fly.

Drawbacks:

- Requires database space and database administrator (DBA) support.
- Large datasets can take a very long time to score.
- New records inserted after batch scoring are not scored.
- Updating the data mining model or scores requires more database and DBA overhead.
- Adding new or changing existing models requires rescoring the data.

Database Does the Scoring

In this approach, data mining features inside the database management system perform the scoring. Several major databases have the ability to score data mining models. The most common approach is to import the predictive model into the database, and then generate scores by using extensions to SQL queries. A key feature of this approach is that the model can be imported and stored in the database. Several standards, such as the Predictive Model Markup Language (PMML), OLE DB for Data Mining, and the JSR-73 Java standards, enable the database to import of predictive models. The sophisticated techniques needed to create the model are not required to score the data. Scoring simply involves mathematical calculations on a set of inputs to generate a result. This approach has the following benefits and drawbacks:

Benefits:

- Scores can be done "on the fly" even if new records are added.
- Updating the model is easier than having to re-score the entire database.
- Requires less database space than scoring the database since scores do not have to be persisted in the database.
- BI applications can take advantage of this approach by using the database's data mining capabilities directly.

Drawbacks:

- Requires database space and database administrator support.
- Requires application knowledge of the database's data mining capabilities. Typically, this is different from the database administration skills.
- BI applications must be customized for each database's data mining implementation.

Business Intelligence Tool Does the Scoring

The third approach for integrating data mining uses enterprise data resources without significantly increasing the database overhead. This is accomplished by importing predictive models into the BI platform as standard metrics. Deploying predictive models in the BI platform allows sophisticated data mining techniques to be applied directly within the business intelligence environment on only the data that has been requested. Like the other approaches, it also has benefits and drawbacks:

Benefits:

- Scores can be done "on the fly" even if new records are added.
- Adding a new model or updating an existing model is simply a matter
- Does not require database space or database administrator support.

Drawbacks:

- Input characteristics need to be passed to the BI application even if they are not displayed on the report.
- Very large datasets may use a large amount of BI resources.

12.2 Data Mining Services in MicroStrategy

MicroStrategy introduced Data Mining Services to provide data mining scoring capabilities in Intelligence Server, making MicroStrategy the only BI platform that supports all three approaches to integrating data mining with mainstream business intelligence. The goals behind the introduction of Data Mining Services were as follows:

- Allow business users to apply the predictive models in their everyday reports, against all relevant data in data warehouse
- Minimize the time between data scoring, and the availability of the score to end users.
- Improve the distribution and reach of the predictive insight through the various user interfaces of the MicroStrategy platform, including the zero-footprint Web client, the integration with Microsoft Office, the Desktop development environment, and the proactive report delivery and user self-service capabilities.
- Provide a unified security and industrial-strength BI architecture for data mining that was integrated with the rest of the BI platform.
- Accomplish all of the above with minimal assistance from the database administrator.

12.3 Data Mining Algorithms

Data Mining Services added numerous scoring algorithms to the extensive library of over 270 analytic functions available in the MicroStrategy BI platform.



Figure 12.2 MicroStrategy supports numerous scoring algorithms that execute the predictive analysis models such as regression, neural network, decision tree, clustering, and segmentation.

- Regression:

  Regression is a powerful, commonly used algorithm that evaluates the relationship of one variable, the dependent variable, with one or more other variables, called independent variables. By measuring exactly how large and significant each independent variable has historically been in its relation to the dependent variable, the value of an unknown dependent variable can be estimated using known values of the independent variables. Regression models are widely used in applications such as seasonal forecasting, quality assurance, and credit risk analysis.

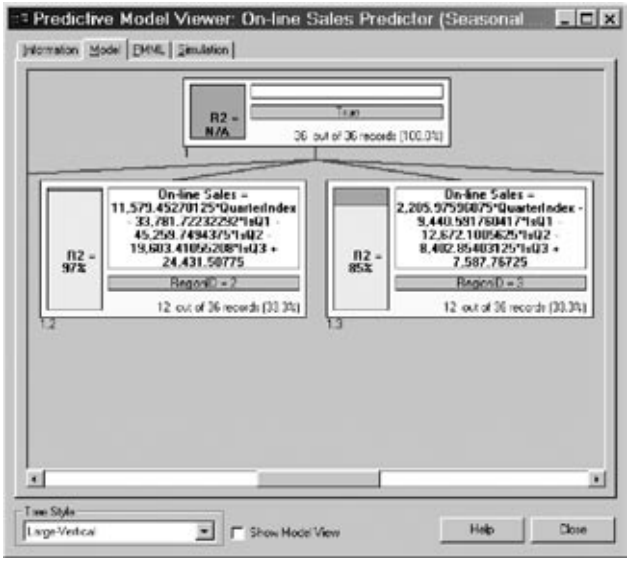


Figure 12.3 An example of a regression model that applies a multi-variate regression predictor for sales by region and quarter. The tree-regression model predicts quarterly on-line sales revenues for each region.

- Neural Network:

  A Neural Network is a sophisticated pattern detection algorithm that uses machine learning techniques to generate predictions. This technique models itself after the process of cognitive learning and the neurological functions of the brain capable of predicting new observations from other known observations. Neural networks are very powerful, complex, and accurate predictive models that are used in detecting fraudulent behavior, predicting the movement of stocks and currencies, and improving the response rates of direct marketing campaigns.

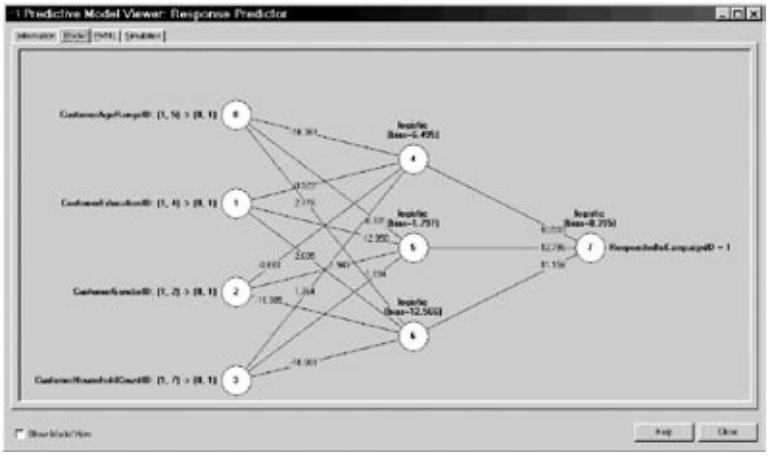


Figure 12.4 Example of a simple neural network model used to predict a customer's likelihood to respond to a marketing campaign using the customer's psychographic profile.

- Decision Tree:

  A Decision Tree is a tree-shaped graphical predictive algorithm that represents alternative sequential decisions, and the possible outcomes for each decision. This algorithm provides alternative actions that are available to the decision maker, the probabilistic events that follow from and affect these actions, and the outcomes that are associated with each possible scenario of actions and consequences. Their applications range from credit card scoring to time series predictions of exchange rates.

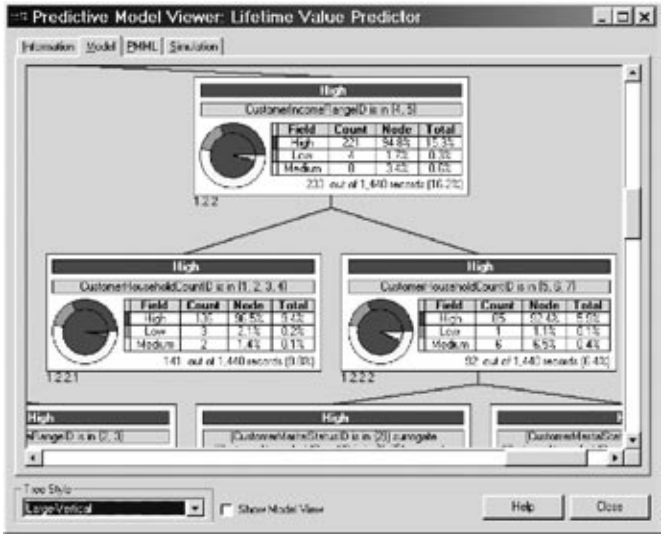


Figure 12.5 A Decision Tree model used to predict the lifetime value of a customer based upon demographic and psychographic factors. It displays the number of customers by income range with a high lifetime value. The tree algorithm then breaks down high lifetime value customers by the number of people in a household.

- Clustering/Segmentation

  Clustering/Segmentation is the process of grouping items together to form categories. Once the characteristics of a group of items are determined, they can be used to find customers with similar characteristics. This algorithm is used to create logical groups, such as customers for marketing campaigns, rate groups for insurance products, and crime statistics groups for law enforcement.

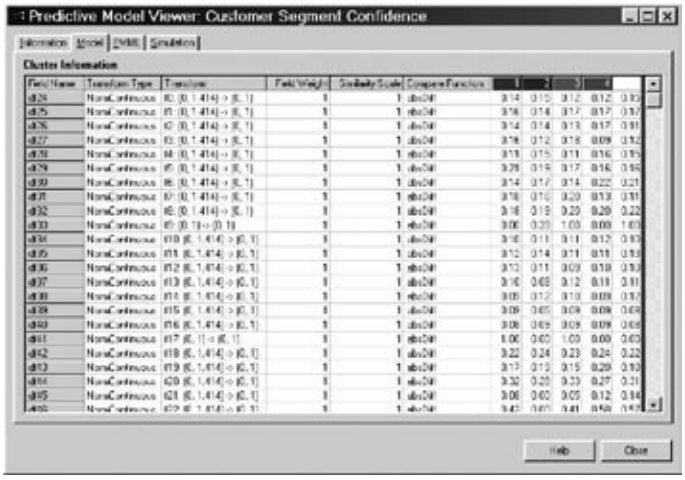


Figure 12.6 Example of a classification model used to segment customers based upon products in their shopping cart.

Predictive Analysis Using MicroStrategy Analytic Library

The MicroStrategy BI platform contains an analytic function library of over 270 basic, OLAP, mathematical, financial, and statistical functions that can be used to create business metrics and key performance indicators. In addition, it offers numerous advanced analysis capabilities such as set analysis, transformations, dimensional, and conditional metrics that can be used in conjunction with the function library to provide predictive analysis without requiring a third-party data mining tool.

For example, market basket analysis identifies customers who are likely to buy certain products based upon the existence of other items in their shopping carts. This technique uses historical purchase information from other customers to identify products that are commonly purchased together. This information can be used to target customers who did not buy a complementary product with a marketing campaign.

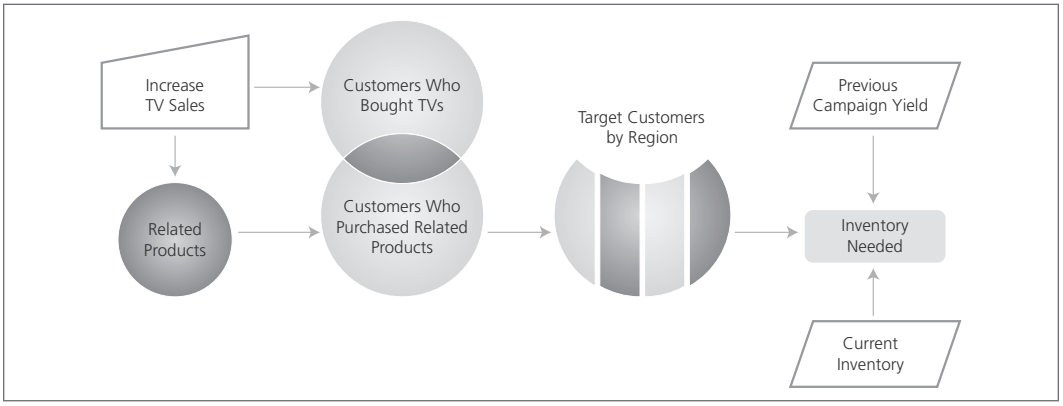


Figure 12.7 Market basket analysis predicts customers likely to purchase TVs.

Another example of the native predictive analysis capabilities in the MicroStrategy platform is the ability to define multi-variate regression models that relate several independent variables to predict an outcome of the dependent variable. The regression model supports both linear and exponential regression. These models can be constructed as tree models where each node of a tree contains a separate regression.

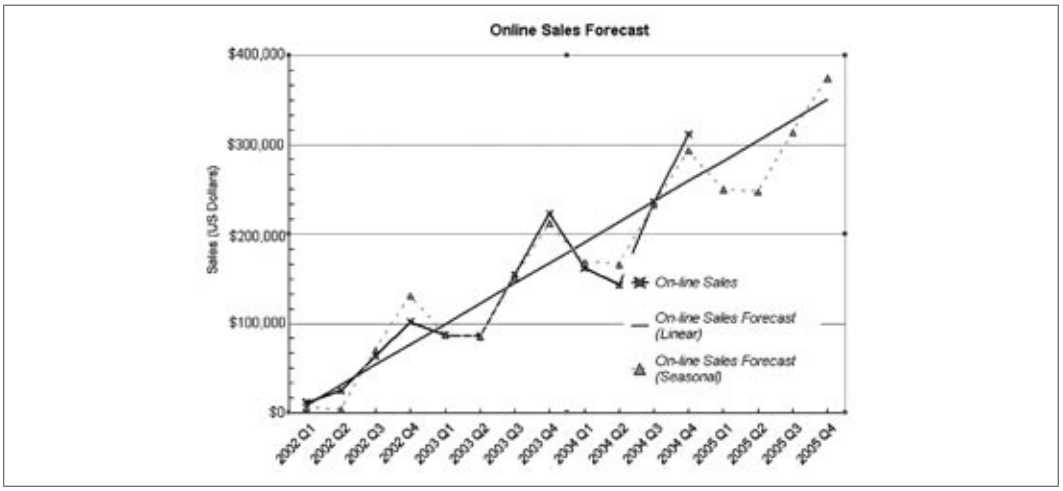


Figure 12.8 Business users can create and view predictive reports just as they would with any other MicroStrategy report. This report shows how linear regression, the straight line, and multi-variate regression, the dotted line, is used to forecast sales using quarterly periods as independent variables.

Regression models are created with a Training Metric Wizard that prompts for the dependent variable, independent variables, and tree levels. The Wizard also specifies the desired outputs of the model, including the score and the confidence level of the score. Upon completion, the wizard generates a standard MicroStrategy metric, and a PMML rendering of the model.

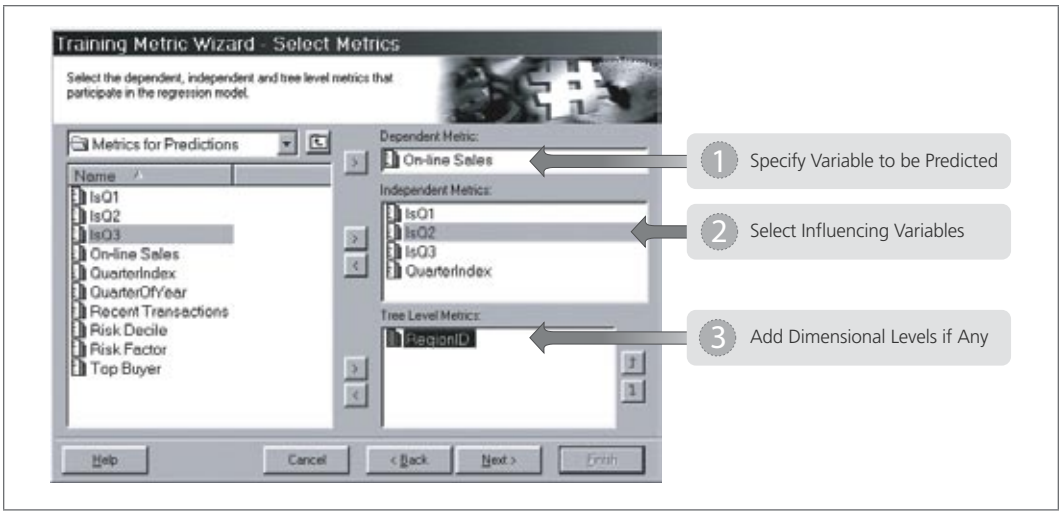


Figure 12.9 A Training Metric Wizard allows users to select the dependent and independent variables, and automatically generates the regression model and associated predictive metric.

Predictive Analysis Using Third-Party Data Mining Products

With Data Mining Services, organizations can maximize their investment in data mining products by importing a predictive modeling mark-up language (PMML) representation of the predictive models generated by the data mining product, and seamlessly encapsulating the model as a predictive metric for use with any report or document. Business users continue to inherit all the object-reuse, security, administrative, and manageability benefits of the MicroStrategy BI platform.

The workflow that integrates the MicroStrategy platform with third-party data mining tools to generate and use predictive metrics is described below.

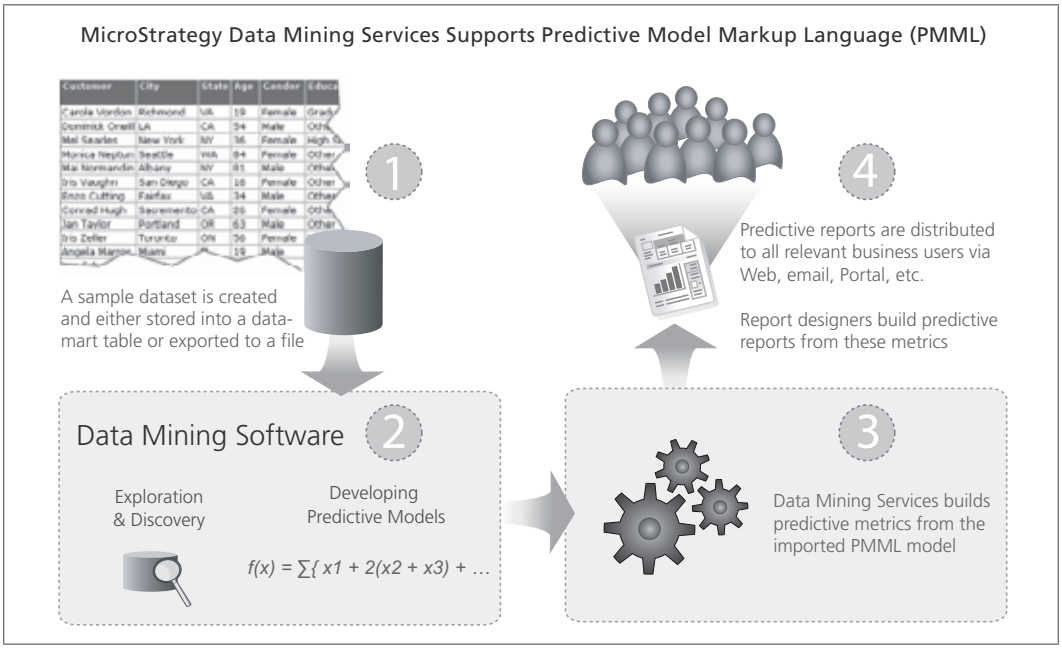


Figure 12.10 Data mining workflow – 1) Create a data sample in MicroStrategy, 2) Import the sample into a data mining tool and develop the predictive model, 3) Import the model into MicroStrategy, and 4) Use the predictive metrics in documents, reports, filters, and other metrics.

1. Create a sample dataset and export it to a third-party data mining tool
   A user creates a standard report in MicroStrategy that contains a known historical outcome to a business issue, and a list of characteristics that may or may not influence the outcome. The user runs this report, and can add more information or drill down or across to other hierarchies to get a desired data sample. Next, the data sample is either exported to various formats, for example text files or Microsoft Excel, or saved to a database table using a dynamic datamart in MicroStrategy. Dynamic datamarts are a preferred approach when dealing with large datasets, whereas exporting to a text file or Excel may be more convenient for smaller datasets.

2. Use third-party data mining tools to develop predictive models
   A domain expert, data mining analyst or statistician imports the data sample into the third-party data mining tool through an ODBC connection to the datamart table or by simply uploading the exported file. The data mining tool analyzes the data sample in search of predictive information, and builds a predictive model from this dataset. When complete, the predictive model is exported to an XML file in PMML.

3. Upload the PMML predictive model, and generate predictive metrics
   The PMML file is uploaded into MicroStrategy, which interprets the predictive model, and automatically creates scoring and confidence predictive metrics. These new predictive metrics are stored in the MicroStrategy metadata, and can be used like any other metric. By complying with the PMML standard, MicroStrategy works with all leading data mining tools.

4. Build and distribute reports with predictive metrics
   Predictive metrics can be added to reports just like any other MicroStrategy metric. All report manipulations, such as sorting, filtering, adding new calculations, and defining thresholds are possible. Predictive metrics also inherit all the object-reuse, security, administrative, and manageability benefits of the MicroStrategy BI platform.

12.4 Benefits of Integrating Business Intelligence and Data Mining

While adding data mining scores and predictive models directly in the database is beneficial, there is additional value to be gained by integrating data mining scoring inside the BI platform.

- Business users can view predictive reports in a wide variety of user interfaces.
- Highly formatted predictive reports provide the easiest possible user consumption and professional presentation.
- Personalized messages and predictive reports can be delivered to very large user populations based on alerts or schedules.
- Ad hoc query and analysis that includes predictive metrics is possible without requiring knowledge of SQL, table structures, or predictive models.
- Business analysts can perform further analysis, such as slice-and-dicing data, ad hoc report creation, drilling, pivoting, and sorting, on predictive reports.
- Strict security is applied to users within, and outside the organization.

The industry leading MicroStrategy BI platform is the first to deliver data mining and predictive analytics to all users through a fully integrated enterprise-caliber BI system. Business users, report designers, and analysts can view and build predictive reports using MicroStrategy, and distribute these reports to all relevant decision makers and stakeholders.

13

Closed-Loop Business Intelligence

Business intelligence reports, dashboards, and analysis provide the foundation for effective, data-driven decision making. One requirement in many business intelligence applications is for managers, analysts, and executives to take immediate action on the data presented. For example, a report showing a low inventory of popular products may contain a mechanism to automatically create purchase orders for these items directly from the BI report. This eliminates the need to manually access other IT applications, or process telephone or paper requests. Closed-loop business intelligence is this interaction with BI reports and other systems, specifically to write data back to any target IT system, and to execute any process necessary to complete a desired action. This closes the loop between the user and the data, turning insight into action.

Closed-loop BI applications vary widely from industry to industry, and even within horizontal applications, such as sales, finance, and human resources. Closed-loop BI applications have a set of common requirements that satisfy a broad set of business demands:

- Reach every business user regardless of location using open, Web technologies
- Connect to any number or type of database-driven applications
- Perform common database functions to modify, insert, or delete data
- Accept and/or direct business logic or workflows to complete intended tasks

13.1 Closed-Loop Applications Reduce the Time to Act

Closed-loop BI applications implemented with the MicroStrategy BI platform reduce the time required to act on business insight, and allow companies to standardize internal processes and procedures like the stock reordering process described above. Furthermore, because business users have this closed-loop BI capability directly within MicroStrategy Web reports, there is no additional training required. The MicroStrategy BI platform implements closed-loop BI using:

- Dynamic Datamarts
- Freeform SQL
- MicroStrategy Transactor

In all cases, MicroStrategy Web is the interface that initiates the closed-loop interactions between the BI application, and any other IT system such as an enterprise resource planning (ERP), customer relationship management (CRM), supply chain management (SCM), data mining or business intelligence application.

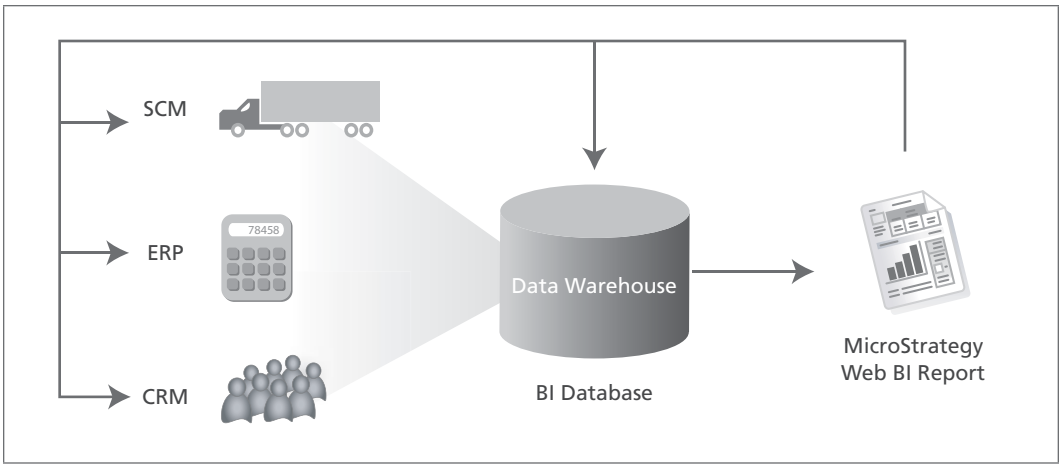


Figure 13.1 A closed-loop BI application turns insight into action by integrating with operational systems, such as SCM, ERP, or CRM applications.

13.2 Closed-Loop BI Using Dynamic Datamarts

Dynamic datamarts create datamarts, or tables of information, in a database using a report definition stored in the metadata. This eliminates the need to move large amounts of data to client machines and export the information. Datamarts are ideally suited for implementing robust, closed-loop BI applications that automate processes for several reasons.

- Creating and executing datamarts in MicroStrategy is straightforward and efficient. MicroStrategy Desktop defines the datamart table structure with a datamart report to distinguish it from other MicroStrategy report types. Once created, MicroStrategy Web or Desktop executes the datamart, populating it with the specified data.
- Datamarting uses MicroStrategy's full breadth of analytical richness including sophisticated filtering, time transformations, and predictive analysis to return the exact set of data required.
- Datamarting benefits from MicroStrategy's optimized SQL generation that ensures datamarts run efficiently and rapidly. Datamarting makes interfacing with the operational systems easy by incorporating pre-SQL and post-SQL capabilities.
- Dynamic datamarting stores the resultant datamart to any available database, including the data warehouse, SCM, ERP, or CRM databases. Once created, these applications use the datamart directly to automate business processes.
- Datamarts are efficient and straightforward to maintain and update. They have the flexibility to refresh entire datamarts or append new data to existing datamarts.

Entire Datasets are Integrated With Other Applications

Consider the scenario of closed-loop application that combines BI and a CRM system. The CRM system contains a wealth of customer-specific data detailing demographic information, purchase history, personal preferences, and much more. MicroStrategy analyzes this data using sophisticated data mining algorithms to understand which customers will have the highest lifetime value, and the highest propensity to respond to a targeted marketing campaign. With the analysis complete, MicroStrategy creates a datamart directly in the CRM system. Using post-SQL scripting capabilities, MicroStrategy informs the CRM system that a new datamart is ready for the campaign management tools to initiate the targeted marketing campaign.

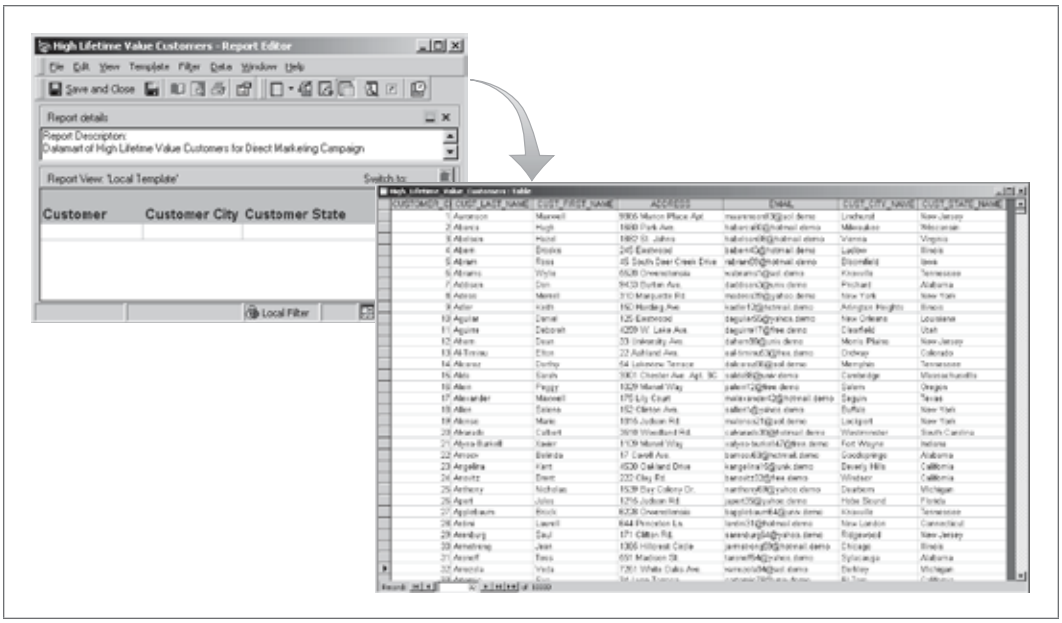


Figure 13.2 The datamart definition (top) lays out fields necessary to create the resulting datamart (bottom) used for the CRM marketing campaign.

13.3 Closed-Loop BI Using Freeform SQL

The single most frequent requirement of closed-loop BI applications is adding to, or modifying, the data in a report, and saving the changes back to a database. This is called database writeback or just writeback. Unlike dynamic datamarting, where entire datasets are created, database writeback requires selective updates to the database, often on a row-by-row basis. MicroStrategy's Freeform SQL capability has the backend flexibility to handle selective writing to the database.

Intelligence Server performs the database writeback using freeform SQL. It provides a middle-tier connection behind firewalls that isolates users from the database ensuring the security of the database. In contrast to using a SELECT statement to retrieve data for reporting and analysis, freeform SQL writes data to any target database using INSERT, UPDATE, or stored procedures. The SQL statement has a great deal of flexibility to achieve a wide range of database actions.

SQL Statement	Action
UPDATE inventory_curr	Update ⇨ SQL command instructing the database to modify a table Inventory_curr ⇨ the table to be modified
SET reorder_qty = [VALUE]	SET ⇨ SQL command instructing the database to modify the reorder_qty column Reorder_qty ⇨ the column name inside the inventory_curr table [VALUE] ⇨ a variable that temporarily stores the user input to be written into column reorder_qty
WHERE item_id = [FORM| Item@ID]	WHERE ⇨ SQL command instructing the database to associate the user input [VALUE] with a particular data element, item_id [FORM|Item@ID] ⇨ a variable that identifies the attribute , in this case the ID

Figure 13.3 An UPDATE SQL statement used to writeback data for the closed-loop BI application broken down into its components with an explanation of the MicroStrategy platform and database interactions

Simple User Interactions on Selected Information

MicroStrategy reports need additional functionality for closed-loop interactions. MicroStrategy Web's extensibility makes incorporating this user interactivity into the interface easy. Standard Web conventions, such as text boxes, pick lists, buttons, and drop-down menus, accept data input from users, and initiate the actual writeback of data to the database.

The MicroStrategy SDK completely documents writeback customizations in MicroStrategy Web using freeform SQL. A "Writeback Editor" sample defines the writeback workflow in four easy steps:

1. Choose the report for writeback.
2. Select the metric used to accept user input.
3. Enter in the appropriate SQL statement for writeback.
4. Choose the control style to accept the user input.

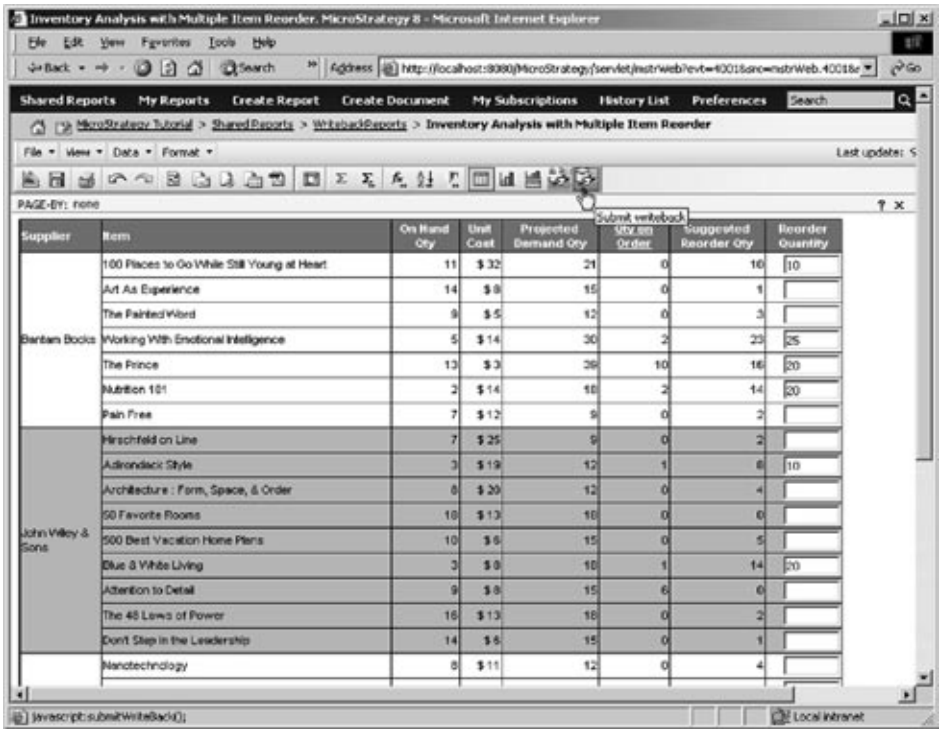


Figure 13.4 A closed-loop BI report in MicroStrategy Web shows low store inventory. The report accepts data inputs for the specified items only, and writes that data back into a Supply Chain Management system, which automatically reorders the low inventory items.

13.4 Closed-Loop BI Using MicroStrategy Transactor

Dynamic datamarting and freeform SQL provide a clean interface point between the BI application and other IT systems because they simply provide data. Such clean demarcations between BI and operational applications do not always exist. For example, a database writeback application may write the same data into two different applications, only if authorized, and both systems verify successful completion. This type of transaction is inherently more complex, and necessitates more flexibility and control. More sophisticated closed-loop BI applications demand the execution of multiple tasks or workflows according to defined business logic or practices.

MicroStrategy Transactor specifically handles these applications, and adds the necessary workflow capabilities to manage these more demanding closed-loop BI requirements. Unlike dynamic datamarting and freeform SQL, which

require a third-party IT system to direct business logic or workflows for closed-loop BI applications, Transactor executes all necessary tasks itself. Transactor is a server that manages all the tasks in a pre-defined workflow to complete a transaction for a closed-loop BI application. For Transactor, a transaction is any sequence of steps or workflow that performs an action in conjunction with one or more IT systems. To manage these more complex workflows, Transactor has a full complement of components to execute, monitor, and design transactions.

Like Freeform SQL, closed-loop reports that use Transactor share all the simple user interactions, such as text boxes and buttons, to capture changes. From a business user's point of view, there is no distinction between the mechanisms in which the closed-loop functionality is implemented.

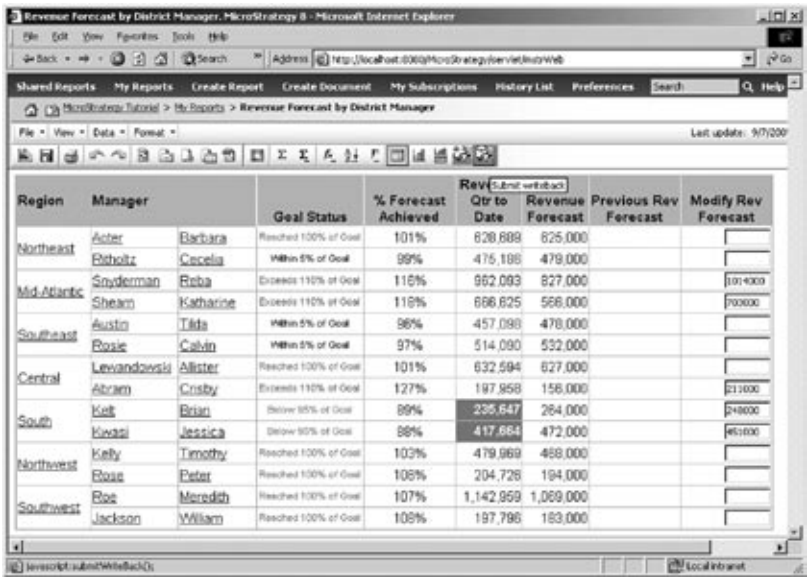


Figure 13.5 A closed-loop BI report in MicroStrategy Web that compares actual revenue to forecasted revenue. MicroStrategy Transactor updates selected forecasts in a Sales Force Automation application and in a Planning application simultaneously.

On the back end, the business logic is more complex when using Transactor than that of the Freeform SQL closed-loop BI option. Transactor typically performs a series of tasks against one or more systems, ensures that each step occurs without errors, and notifies the user that the tasks have completed successfully. If any of the steps result in an error, a separate branch of the workflow rolls back all the changes, and notifies the user of the error.

These additional steps require precise handling to execute successfully. Transactor must not only communicate with the systems but also must retrieve additional pertinent information from the systems, process, and transform the information, and communicate the results to those systems and the user. The following sections explain how the Transactor architecture, intercomponent processing flows, and workflow design achieve this result.

Transactor Architecture

MicroStrategy Transactor is a workflow server with a full complement of components to design, execute, and monitor transactions. Transactor processes all information internally as XML. It accepts data from any hypertext source, such as a Web browser or PDA, transforms that data accordingly, and communicates with other back end systems. The major components of MicroStrategy Transactor are:

- Gateway – interacts with the business user, and converts client transaction requests into XML format
- Communications Service – queues transaction requests, assigns Transaction Engines to execute the requests, and performs load distribution and failover

- Transaction Engine – executes all the steps required to manage and conduct a transaction
- Drivers – communicate with external systems, including databases, legacy systems, and ERP systems. Transactor has a database driver for interacting with databases, a Java Message Service (JMS) driver for interacting with enterprise messaging products, such as IBM® MQSeries® that support JMS, and a Java API for creating custom drivers.
- Logger – records detailed information about each step in the transaction execution process
- Administrative Console – configures and monitors the Transactor system
    - Views real-time status of the Transaction Engines
    - Monitors system activity during transaction processing
    - Controls the Transaction Engines
    - Configures Transactor system components, security, and event logging
- Transaction Designer – creates the Transaction Definition through a simple graphical interface. The Transaction Definition is an XML file that contains the business logic modeled into a sequence of steps or workflow.

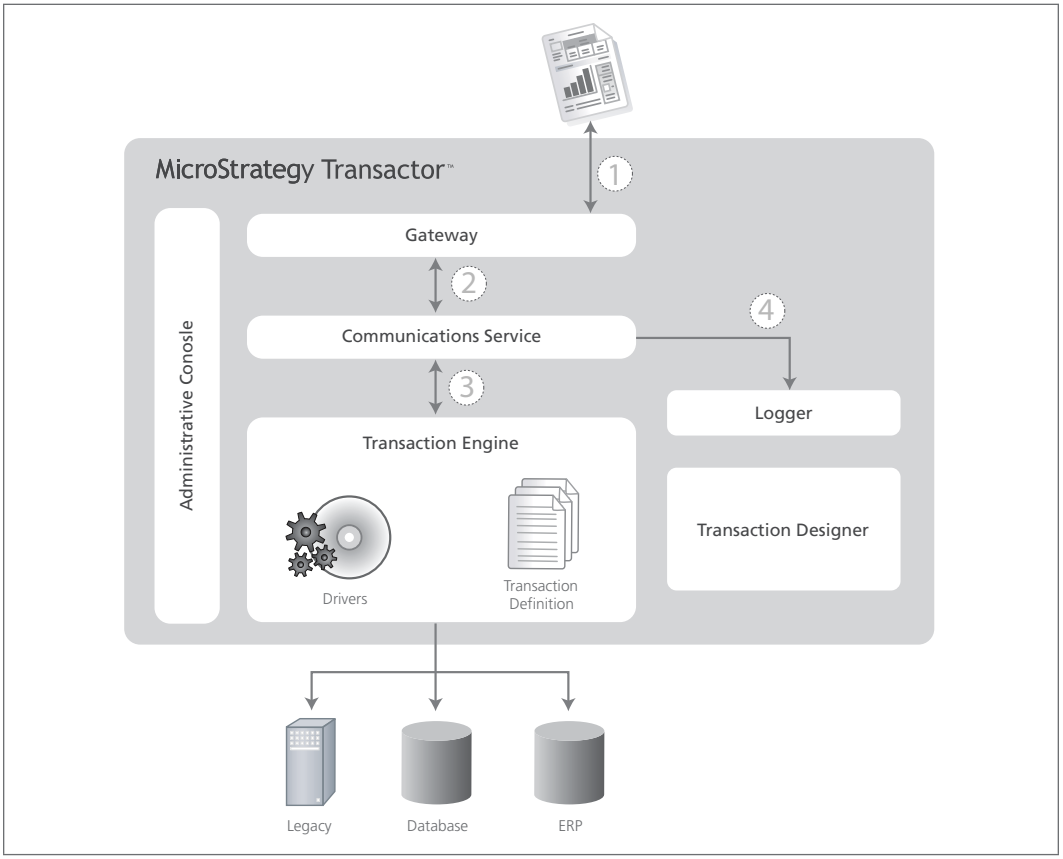


Figure 13.6 MicroStrategy Transactor architectural components and process flow.

The workflow of a closed-loop request through Transactor is as follows:

1. A business user initiates a closed-loop transaction from a report. The Gateway receives the request, and translates any key data elements into XML data. The Gateway also maintains sessions with the business user.

2. The Communications Service receives and queues the transaction request. It manages communications between the Gateway, Transaction Engine, and Logger, maintaining and passing data as needed.

3. The Transaction Engine manages the following steps in the workflow to execute a transaction:

    i. Requests XML information from the Communications Server as needed
    ii. Retrieves transaction definition to begin transaction execution
    iii. Acquires the necessary drivers for interacting with target systems, such as databases and ERP systems
    iv. Performs any required transformations on the XML information, such as a conversion from XML to SQL, directs, monitors, and evaluates the results of every step of the transaction until successful completion.
    v. Maintains contact with the business user where necessary through data posts to the Communications Service.
    vi. Generates status information from every step of transaction, and passes that information to the Logger via the Communications Service.

4. The Logger records information about each step in the transaction, such as errors, warnings, and successful completion to a database for tracking, auditing, and monitoring.

Many Transaction Engines working in parallel provide load balancing and failover capabilities to support scalable, 24x7 operations. The Administrative Console configures each Transaction Engine, which can operate on separate server machines, in the Transactor system. For load balancing, the Communications Service assigns jobs to the Transaction engine under the lowest stress. In the case of a Transaction Engine failure, the Communications Service reroutes lost jobs and critical in-process transaction information to an available Transaction engine.

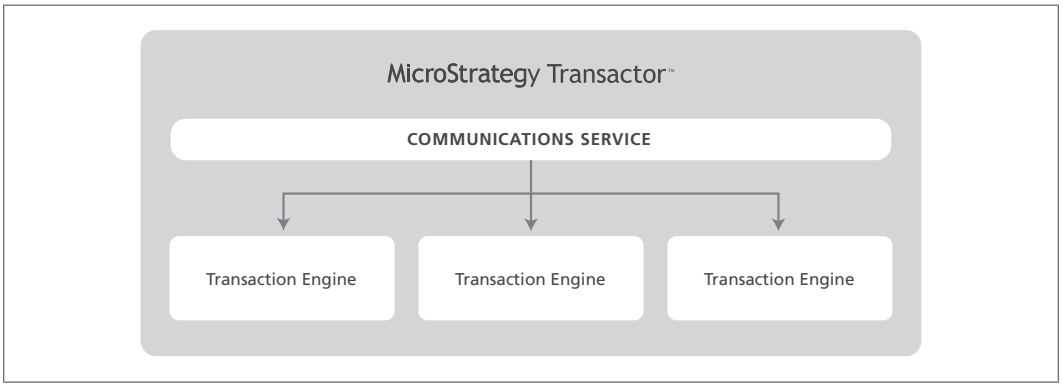


Figure 13.7 MicroStrategy Transactor offers scalability, load balancing, and failover support by adding Transaction Engines to the system.

Transaction Workflow Model and Design

For Transactor, a transaction is any sequence of steps, or workflow, that performs an action in conjunction with one or more IT systems. Transactor models these tasks as a logical flow of steps based on a Petri-net mathematical model. The figure below shows a model of a transaction that updates an ERP system and an enterprise data warehouse:

- Receive data input from a report
- Check user privileges on the ERP and BI systems
- Verify change history to track recent changes and resolve conflicts
- Write the data to the ERP and BI databases simultaneously
- Restore the original data if errors occur
- Display the results of a successful transaction

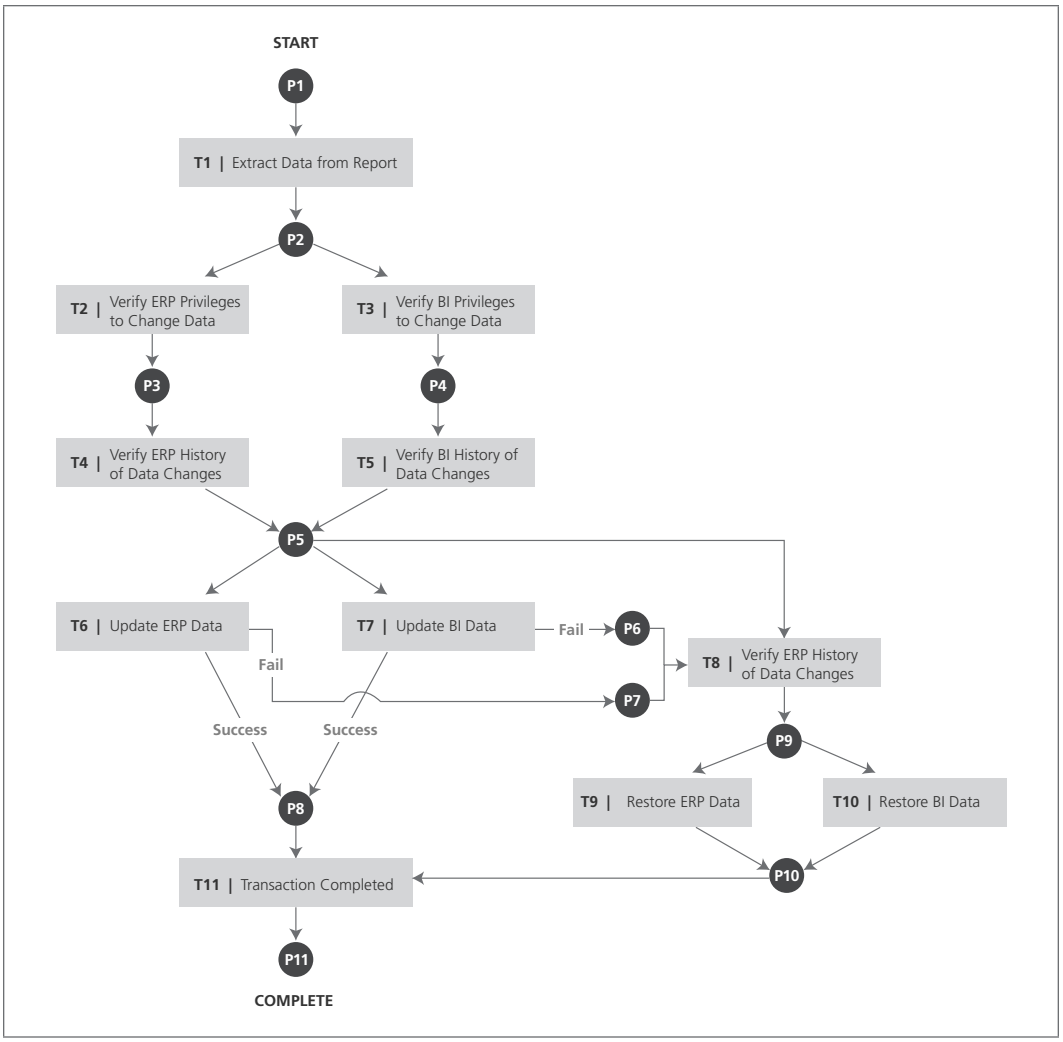


Figure 13.8 MicroStrategy Transactor designs and executes the workflow as a sequence of steps based on a Petri-net Model

The Petri-net model uses three key elements to describe the workflow and the actions that take place at each location.

1. Place – holds the transaction data at some point in the workflow. It is either enabled or disabled. They are represented by circles with ids such as "P1."
2. Transition – accomplishes a discrete task, such as transforming data from XML to SQL, accessing non-MicroStrategy IT systems, or writing data to a database. They are shown as rounded rectangles with ids such as "T1."
3. Arc – joins Places and Transitions to define the logical flow. Arcs also perform decision making by choosing the logical path to follow based on the data stored in the previous state.

This model provides flexibility and sophistication to solve the easiest or most complex of tasks. There are four distinct workflow types:

- Sequential – one step proceeds after the other
- Parallel – two or more logical branches execute simultaneously
- Conditional – one branch executes depending on the data value in the previous state
- Resolution – all converging branches must complete successfully before the transaction continues

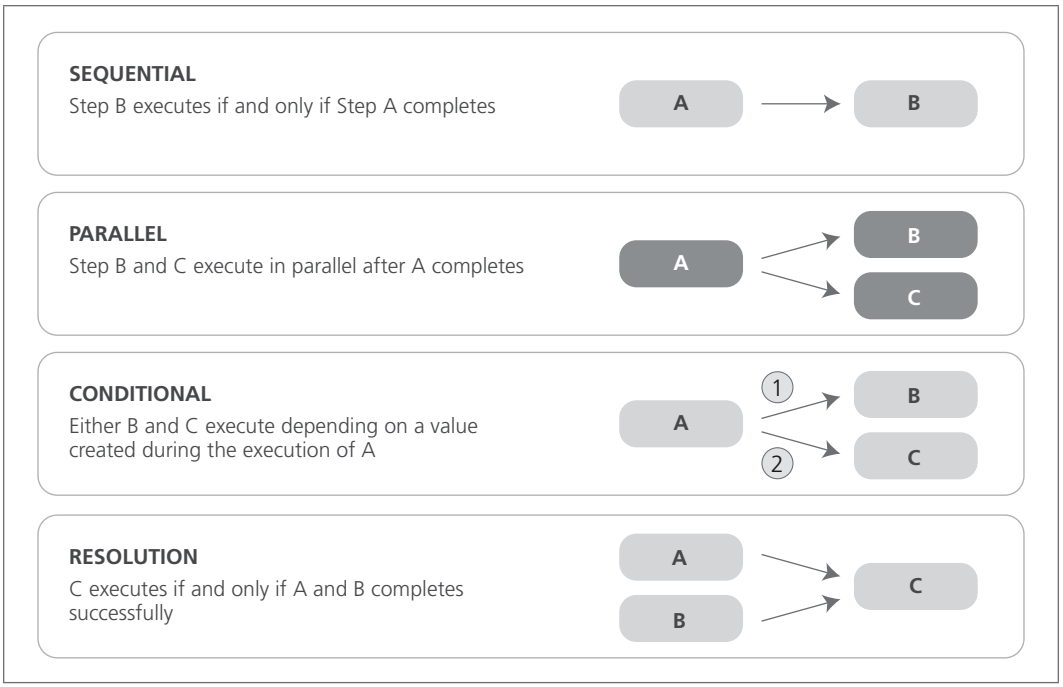


Figure 13.9 The workflow steps execute in any combination, maximizing transaction design flexibility.

The Transaction Designer is a graphical user interface that creates a Petri-net workflow model. It not only represents the necessary workflow logic, but also translates that graphical model into the actual XML transaction definition used by the Transaction Engines.

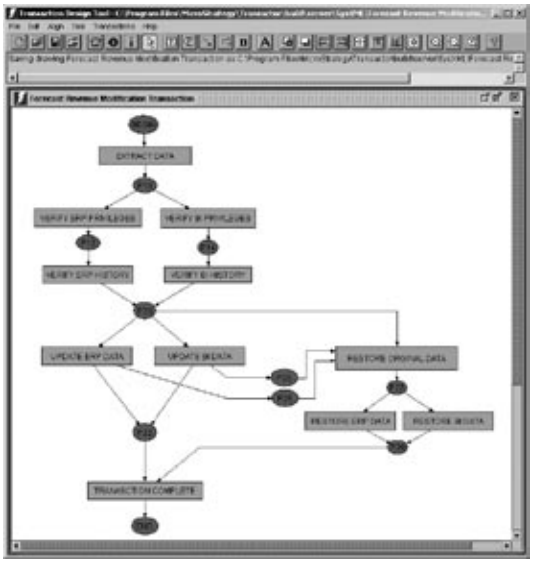


Figure 13.10 The Transaction Designer definition for the example of modifying forecasted revenue in an ERP system and BI database.

13.5 Summary

Closed-loop business intelligence describes the ability to interact with data, specifically to write data to any target IT system, and to execute any process necessary to complete a desired action. This closes the loop between the user and the data, turning insight into action. Closed-loop BI applications reduce the time required to act on business insight. The MicroStrategy BI platform delivers three powerful options to cover the range of closed-loop BI applications using dynamic datamarts, freeform SQL, and MicroStrategy Transactor. Each one has specific strengths depending on the application's requirements.

Datamarting is particularly effective in generating large lists of data that get stored in a database as datamart tables. This provides a carefully chosen set of data for use by other applications.

Freeform SQL is outstanding for database writeback. Changing the data on a report, and saving that change to the appropriate database is the most requested closed-loop BI application type. Freeform SQL handles this superbly, giving the business user a simple Web experience for changing data directly through a MicroStrategy Web report.

MicroStrategy Transactor delivers the flexibility to implement complex closed-loop BI applications. Unlike dynamic datamarts and freeform SQL, which may require third-party IT systems to direct the final steps of closed-loop BI applications, Transactor manages all required tasks. Transactor is a workflow server that manages a sequence of steps, usually in conjunction with one or many other IT systems, to complete the closed-loop BI application.

USER INTERACTIVITY

14

End User Experience

The end user experience with a BI technology is often the most important factor in determining the success of a Business Intelligence application. Business Intelligence applications are now commonly used by all users within an organization – from executives to casual business users. This has led to the growth of three distinct sets of user requirements.

The MicroStrategy BI platform has been architected from the ground up to be intuitive for the business user. With consistent interfaces spanning Web, Microsoft Office, and Windows paradigms, MicroStrategy's user interfaces require minimal end user training, and lead the industry in terms of end user adoption. Users can move seamlessly between various styles of BI applications without switching interfaces, leading to high productivity. MicroStrategy's end user interfaces are also heavily self-service-oriented, allowing even casual business users to format and personalize report content. This characteristic drastically reduces iterative interactions with IT developers.

1. A Business Intelligence platform that provides a consistent, unified, and intuitive user experience to business users
2. The ability for users to transition seamlessly between reporting and analytics without switching interfaces or paradigms
3. The need for a self-service architecture allowing users to create, modify, and subscribe to reports

In this chapter, the various end user interfaces in the MicroStrategy BI platform are first studied. Next, the user roles in an organization, and how MicroStrategy's interfaces can be tailored for each of these user roles, are examined. Finally the 5 styles of BI applications, and how end users perform interactive reporting and analysis within these applications are discussed.

14.1 Consistent Reports, Functionality, and Security through Many User Interfaces

The MicroStrategy BI platform provides a wide variety of user interfaces that support all levels of users for any business intelligence application. Since all of the interfaces share the same metadata, users can move from one interface to another without any re-configuration or administrator intervention. A report created over the Web can be viewed in Windows, Office, and in emails without the loss of any functionality, security, or object privilege settings.

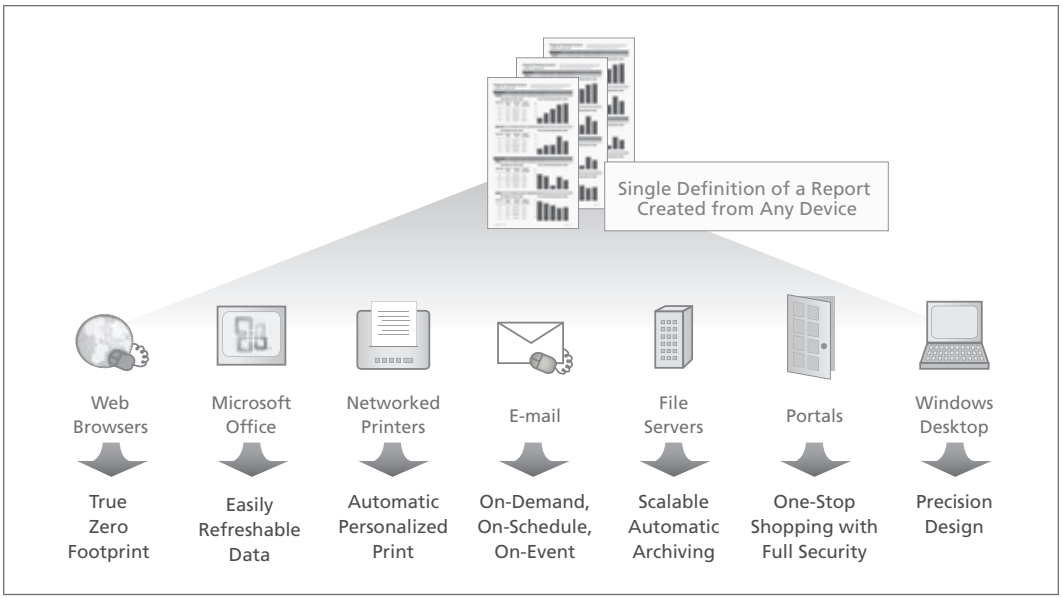


Figure 14.1 Reports are expressed identically through various MicroStrategy interfaces.

The MicroStrategy BI platform architecture includes three primary interactive user interfaces.

1. MicroStrategy Web
2. MicroStrategy Desktop
3. MicroStrategy Office

In addition, the platform includes MicroStrategy Narrowcast Server, which is a distribution interface for email, print, file, and portable device distribution.

Interface	Microstrategy Product
Any Web Browser	MicroStrategy Web
Windows	MicroStrategy Desktop
Microsoft Excel	MicroStrategy Office
Microsoft PowerPoint	
Microsoft Word	
Microsoft Outlook	
Email	MicroStrategy Narrowcast Server
PDA	
Cellular Phone	
Network Printer	
File Server	

Figure 14.2 The MicroStrategy BI platform interacts with a number of applications and devices.

One of the fundamental design tenants of the MicroStrategy platform has been to ensure that any report can be run through any interface, and have the same look and feel. In addition, user actions such as pivoting, page-by, sorting, and drilling are available across all interfaces. As a result, a user can seamlessly transition from one interface to the next with minimal training. The next sections discuss the various MicroStrategy user interfaces in more detail.

MicroStrategy Web

The Web browser is increasingly being adopted as the interface of choice for enterprise applications. MicroStrategy Web is the easiest-to-use BI interface in the industry, and was designed to be highly intuitive. MicroStrategy introduced the first interactive Web interface for business intelligence applications in 1996, and today provides the most comprehensive set of end-user functionality over the Web. In fact, few business intelligence products, irrespective of interface, can match the versatility of MicroStrategy Web in terms of report creation, interactivity, formatting options, data manipulation, and ability to integrate with third-party applications.

MicroStrategy Web is suitable for all levels of users from high-level executives, to power users and analysts, to information consumers. All of this functionality is provided through a browser-independent, zero-footprint Web interface that does not use ActiveX components or Java applets.

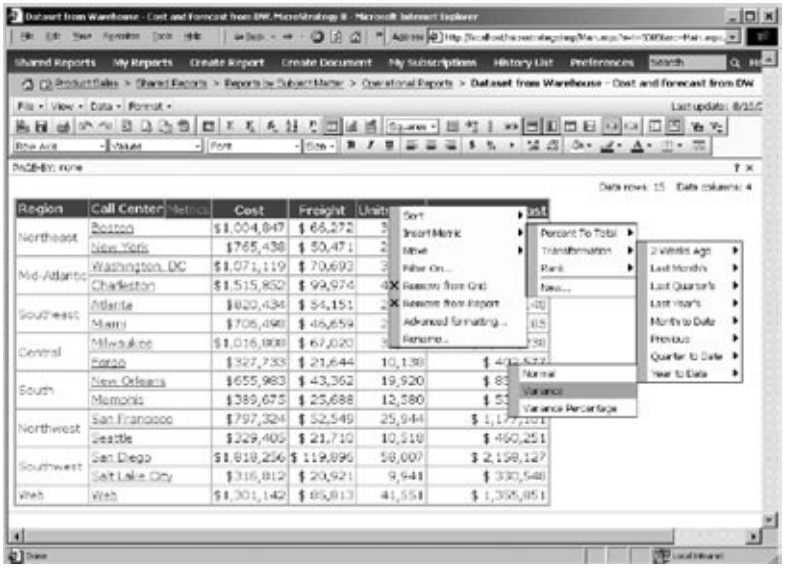


Figure 14.3 Complete report interactivity is available in MicroStrategy Web.

MicroStrategy Web is the most widely deployed MicroStrategy interface, with over 80% of MicroStrategy users accessing their reports and analytics through a Web browser. It has the highest user adoption rate in the BI industry for a number of reasons:

- MicroStrategy was the first BI vendor to introduce a full-function BI Web interface in 1996. This early introduction of the Web paradigm has allowed MicroStrategy to refine Web interactions, and make interfaces ideally suited to end users.
- MicroStrategy Web was rebuilt with the release of MicroStrategy 7 to be the first zero footprint multi browser Web interface. The lack of client dependencies makes it easy to deploy applications to large user populations. Thin client interfaces ensure that security considerations with firewalls are not an issue. MicroStrategy Web achieves a stateless connection without the use of cookies, further simplifying deployments in cases where corporate security policies do not allow cookie storage on user machines.
- MicroStrategy Web is based on the latest "AJAX" techniques, allowing for an unprecedented level of Windows-like user interactivity through a Web browser. Windows on the Web interactivity includes drag-and-drop, dropdown menus, context sensitive right click actions, mouse over tool tips, and undo/redo actions. As a result, MicroStrategy Web can instantly be used by anyone familiar with internet browsing and Windows application paradigms.

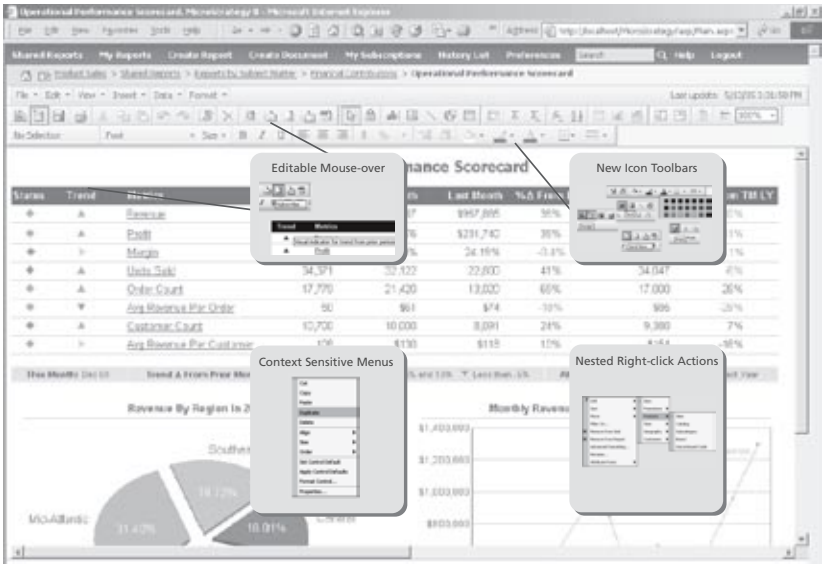


Figure 14.4 Interface optimized for the business user.

MicroStrategy Desktop

MicroStrategy Desktop provides a powerful environment for creating BI applications, building the business abstraction layer, designing and running reports and documents, and monitoring and administering the BI environment. End users can build sophisticated reports, modify them on the fly, and drill anywhere for train-of-thought analysis. The reporting flexibility of MicroStrategy Desktop allows users to change any parameter of a report at any time during analysis, and save any portion of the result – the complete report, the report filter, or the report template. Users can also combine report filters to generate sophisticated qualifications for set analysis or custom groups. Power users can develop and deploy customized workflows and drill paths for navigating through scorecards, dashboards, and other analytical reports.

MicroStrategy Desktop is the primary designer and power user interface in the MicroStrategy platform. It is a Windows-based application that offers full system architecting, application design, system configuration, and administrative functionality.

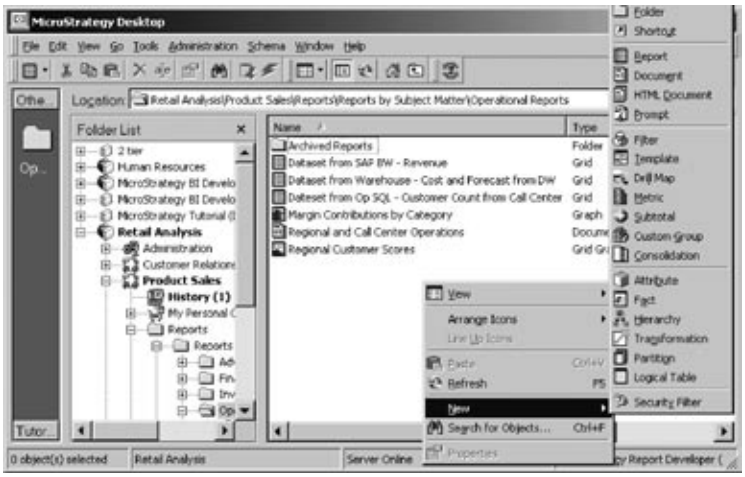


Figure 14.5 Powerful analysis with MicroStrategy Desktop.

MicroStrategy Office

The MicroStrategy Office interface is designed for business users who are already familiar with Microsoft Excel, PowerPoint, Word, and Outlook, and need to be able to insert business intelligence into their Microsoft Office files. Once a report is inserted into a file, it can be refreshed from the file itself. All formatting, prompting, security, and other features of the MicroStrategy BI platform are automatically applied, and the other parts of the Microsoft file are preserved. As a result, users can take advantage of the best functionality available in both MicroStrategy and Microsoft Office.

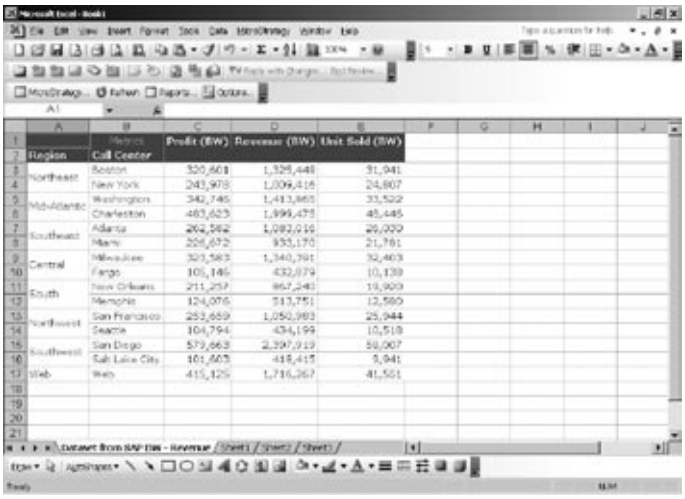


Figure 14.6 BI Report in Excel through MicroStrategy Office.

MicroStrategy Narrowcast Server

In addition to the three primary end user interactive interfaces mentioned above, the MicroStrategy BI platform features a distribution/subscription interface that delivers information to user-specified devices on an event or time-driven schedule.

MicroStrategy Narrowcast Server distributes reports and documents via email or to network printers and file servers. These reports can be delivered on a periodic schedule or when an alert condition is triggered; for example, they can be delivered every Monday morning, at the end of the month, or when the inventory for a particular product falls below a predetermined level. Users can subscribe to reports, specifying whether the reports are personalized, and to which email accounts or other devices they should be sent. Alternatively, an administrator can schedule report distributions. The common metadata architecture of the MicroStrategy BI platform makes reports that are created in either the Web or the Windows environments immediately available for delivery and subscription in Narrowcast Server.

MicroStrategy Narrowcast Server can distribute information to user groups ranging from 10's to 1,000,000's of people while simultaneously personalizing the information delivered to each recipient according to their preferences and security profiles.

14.2 The Business Intelligence Application Spectrum

As business intelligence has evolved from hard-coded back office applications to pervasive enterprise deployments, user functionality has become an essential consideration. Over the past five years, many companies have acquired a diverse collection of business intelligence technologies because these different technologies offered a particular strength in a single area. Broadly speaking, business intelligence applications can be grouped into five application areas, or styles of BI:

1. Scorecards and Dashboards – Highly graphical reports designed to monitor corporate performance
2. Reporting – Print-perfect operational and business reports with interactive content
3. OLAP Analysis – Slice-and-dice analysis with drilling, pivoting, and other investigative features
4. Advanced Analysis – Fully investigate queries with set analysis, statistical and trend analysis, and data mining
5. Alerts and Proactive Notification – Continuous scanning of data for exceptions and anomaly-based reporting

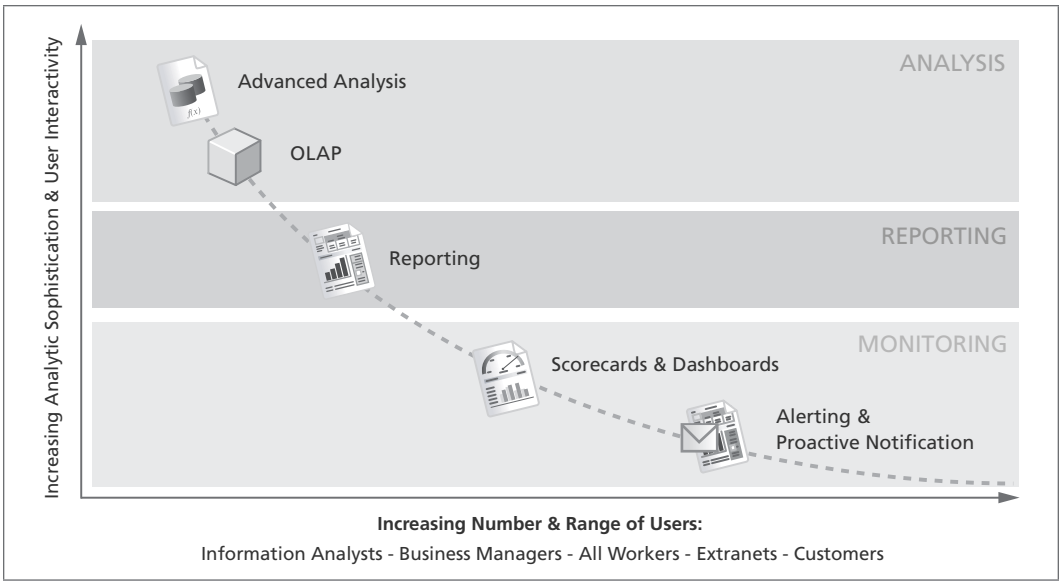


Figure 14.7 Over the preceding decade, BI vendors have produced various different technologies to support the 5 styles of BI applications. Only MicroStrategy delivers all 5 styles of BI within a single unified architecture.

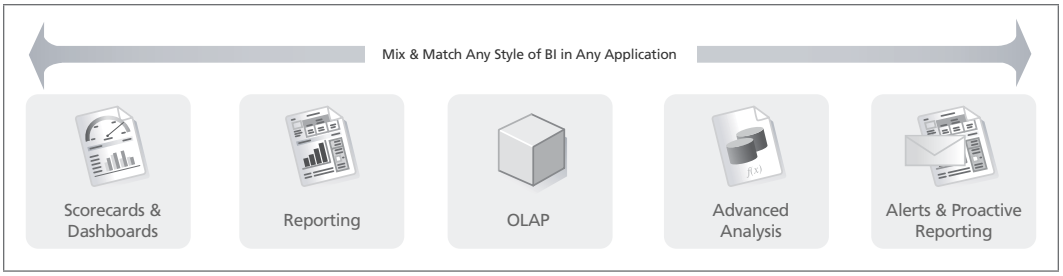


Figure 14.8 Users move seamlessly between any and all Styles of BI in any MicroStrategy interface.

Traditionally, organizations have had to maintain multiple different toolsets to provide users with functionality across these application areas. The MicroStrategy BI platform gives users a single platform and interface from which to access all of these applications, and to navigate seamlessly between different application areas. For example, users can view a scorecard formatted for viewing and printing, drill from the scorecard to obtain more information from an OLAP cube, perform advanced statistical analyses, and collaborate with other users. All of these steps are achieved without the users having to switch interfaces or applications, allowing them to maintain context, and move data between applications.

14.3 MicroStrategy User Experience Design Tenets

The MicroStrategy BI platform was designed specifically to eliminate the need for organizations to maintain multiple disparate architectures for BI. The design tenets for MicroStrategy's BI user functionality were simple:

1. Provide the complete range of business intelligence functionality in one platform
2. Design interfaces that are intuitive to the business user.
3. Provide interfaces tailored to user needs by allocating BI functionality on a user, user role or user group level without customization or coding.
4. Allow users to seamlessly acquire additional BI functionality as their experience and needs increase.
5. Offer BI functionality uniformly through all popular interfaces – Web, Windows, Microsoft Office, and email.
6. Deliver Web functionality through a pure HTML/DHTML browser – without using ActiveX or Java applets – to ensure airtight security and scalability to millions of users.

The MicroStrategy BI platform is the crowning achievement of these goals. On a single platform, users can access multiple styles of BI and seamlessly move from style to style in the same interface. In addition, as users demand more functionality, the BI interface is automatically adjusted from simple to advanced to fit their BI requirements and permissions.

14.4 Scorecards and Dashboards

Scorecards and dashboards are used to measure and monitor operational performance by presenting key measurement indicators from various areas on a single screen in an easily understandable manner. MicroStrategy Report Services enables users to view, edit, create, and modify scorecards and dashboards.

Casual business users can view, execute, export, and send these scorecards to other users. Users can also edit these documents in a WYSIWYG manner to alter content and formatting. Scorecards and dashboards within the MicroStrategy BI environment are not static documents, but are interactive for the user. Users can perform OLAP-type manipulations such as sorting, pivoting, and drilling on these documents.

Management Dashboard applications monitor key performance indicators across the enterprise, and are often synonymous with high-level scorecards. MicroStrategy provides an ideal platform for deploying management dashboard applications. Users can create and view presentation quality documents consisting of information from multiple reports. Additionally, each report can be formatted to the exact specifications of each user. MicroStrategy provides users with important features such as proactive alerting and the choice of multiple interfaces, including Web, Windows, email, and Excel.



Figure 14.9 Dashboard to monitor operational metrics.

Scorecard applications leverage advanced analytics that include metrics calculated at the atomic level of data stored in large data warehouses. In addition, to providing a high level overview of operations, scorecarding applications need to identify cause and effect. MicroStrategy products allow users to view scorecard reports at the required depth and breadth of analysis. Moreover, interactive ad hoc query and report manipulation capabilities allow users to perform train-of-thought analysis, or analyze a report to its logical conclusion.

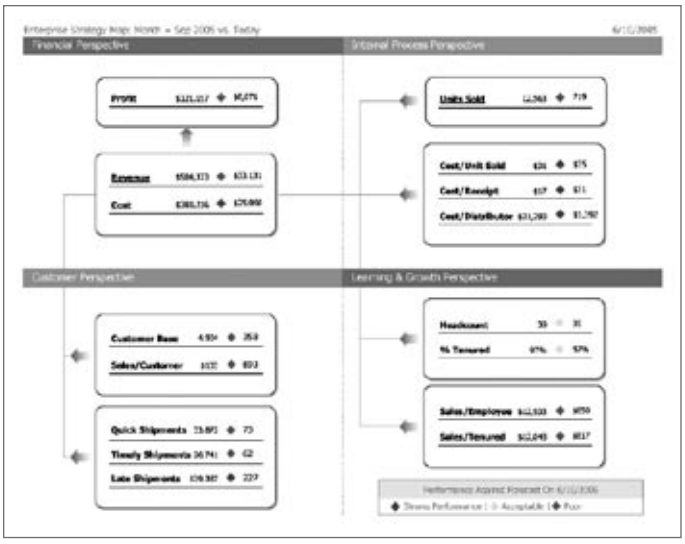


Figure 14.10 Balanced Scorecard application with a strategy map.

Scorecards and dashboards are used in corporate performance management and executive dashboard applications. These reports quickly identify when key performance metrics fall outside a desired range, and how they track over time. The most detailed scorecard and dashboard reports contain many metrics to gain deeper insight into the status of a business process. Users can use MicroStrategy's drill anywhere capability to fully investigate the data, and discover why a particular indicator has fallen outside an acceptable range.

Department Operational Metrics For Week Ending 11/21/03

11/24/2003 8:57 AM

Status	Trend	Metric	Actual	Target	% Difference
●	▲	Revenue	$756,458	$875,000	-14%
■	▲	Operating Expense	$286,893	$200,000	43%
◆	—	Gross Profit Margin	41%	65%	-37%
●	▲	# Unique Prospective Buyer Visits	227	250	-9%
●	▲	Lead-To-Order Conversion Rate	15%	25%	-40%
◆	—	% Sales To Total Region Sales	30%	40%	-25%
■	▲	# Open Sales & Service Positions	5	3	67%
■	▲	Average Employee Turnover	34%	25%	36%
◆	▼	Average Employee Performance Score	3	5	-40%
●	▲	Sales Of Existing Models	$385,794	$400,000	-4%
●	▲	Sales Of New Models	$226,938	$396,000	-43%
●	—	Sales Of Pre-Owned Models	$143,727	$80,000	80%
●	▲	# New Customers	47	50	-6%
◆	▼	% Customers Returning For Service Within 6 Months	72%	90%	-20%
●	—	Average Customer Satisfaction Score	5	5	-3%
◆	▲	Regional Ad Campaign Expense	$750,000	$750,000	0%
●	—	Quality Level Associated With Brand	5	5	-1%
◆	▲	% Closed Deals With Incentives Applied	81%	59%	62%

Figure 14.11 Scorecard showing trends for various key figures.

Unique MicroStrategy Strengths for Scorecards and Dashboard Applications

- WYSIWYG Formatting for All Users

  MicroStrategy Web extends report design capability beyond specialized report developers to include business users. For the first time in the industry, business users can design and refine reports in a full WYSIWYG (What-you-see-is-what-you-get) view through a zero-footprint Web interface. WYSIWYG design uses the innate skills needed to design or refine a MicroStrategy report that most business users already possess from their experience with Microsoft PowerPoint or Excel.

  This provides a radically simpler model for developing reports, and makes it possible for average business users to "tune" reports to their exact structure and formatting needs without time-consuming iterations with professional report designers. With MicroStrategy Web's WYSIWYG design, available to both business users and report developers, companies achieve radically faster time-to-deployment for new scorecards, dashboards, and enterprise reports. Report developers can design reports quickly, and then send them to business users to add context and details that make the report most meaningful. This saves time for report developers and business users, freeing both to perform their primary job functions most efficiently.

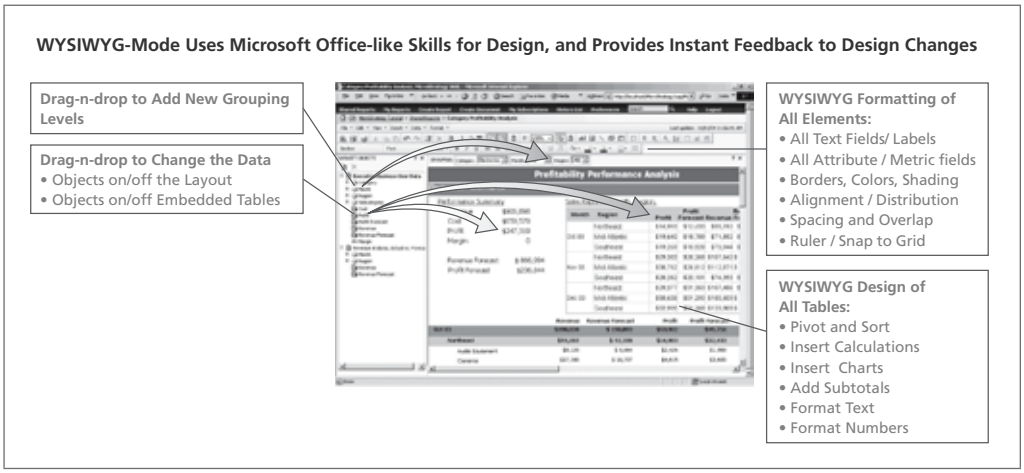


Figure 14.12 Full control of the report layout with WYSIWYG design.

- Fully Interactive Reports and Scorecards

  MicroStrategy provides full interactivity for all reports and scorecards. Business users can rearrange the layout, formatting, and content of any report with simple drag-and-drop actions or by clicking on toolbar icons to get entirely new views of the data without requiring assistance from IT.

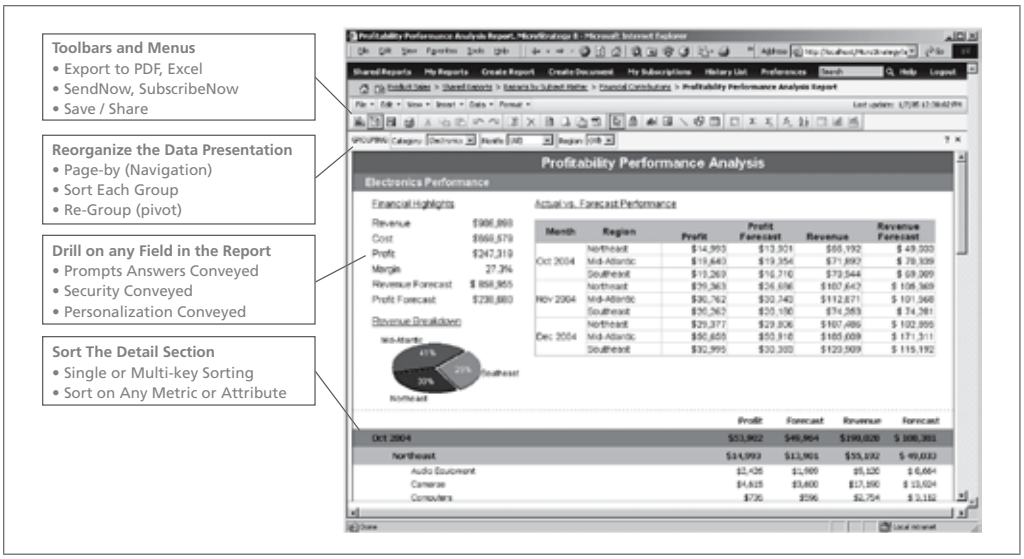


Figure 14.13 Interactive reporting through MicroStrategy Web.

14.5 Reporting

Reporting Applications provide users with print perfect formats that deliver detailed information as operational reports, business reports, invoices, and statements. Standard operational reports are used extensively in all types of analytical applications. Standard reports are the primary reports that are available in any reporting application. Users often customize these reports to meet their own specifications. The MicroStrategy BI platform provides the highest degree of flexibility in analyzing standard reports, so that a user can tailor a report view to the exact level of information and format required. Sophisticated caching within the MicroStrategy BI platform ensures that users quickly receive their most complex reports at the most granular level of detail necessary.

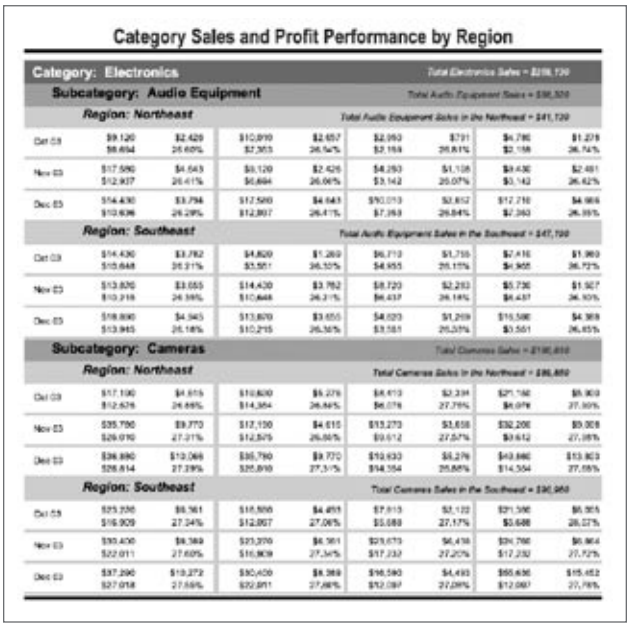


Figure 14.14 Densely populated operational enterprise report.

Unique MicroStrategy Strengths for Reporting Applications

- Powerful WYSIWYG Formatting and Layout Options
  The MicroStrategy BI platform provides extensive formatting and layout features to ensure that the presentation of reports and scorecards satisfies end user and corporate specifications. Out-of-the-box data formatting options include:

  - Font – Size, font face, color, bold, italics, underline
  - Number format – Decimals, currency, number of digits, date formatting
  - Thresholds and indicators – Conditional formatting, stop-lights, gauges
  - Grid formats – Borders, cell background, shading for headers, headers format, alignment
  - Graph formatting – Number of series, legends, axis, title, series along rows or columns, scale, fonts, graph types
  - Text boxes, tables, images – Add text boxes, and insert static images into documents and scorecards
  - Outline mode – Collapse and expand reports at various levels for easy aggregation by clicking on a single button
  - Zone-based reports – Freeform layout of scorecards and dashboards with images, text, grids, and graphs
  - Banded reports – Hierarchically organized reports and scorecards with banded groups of information

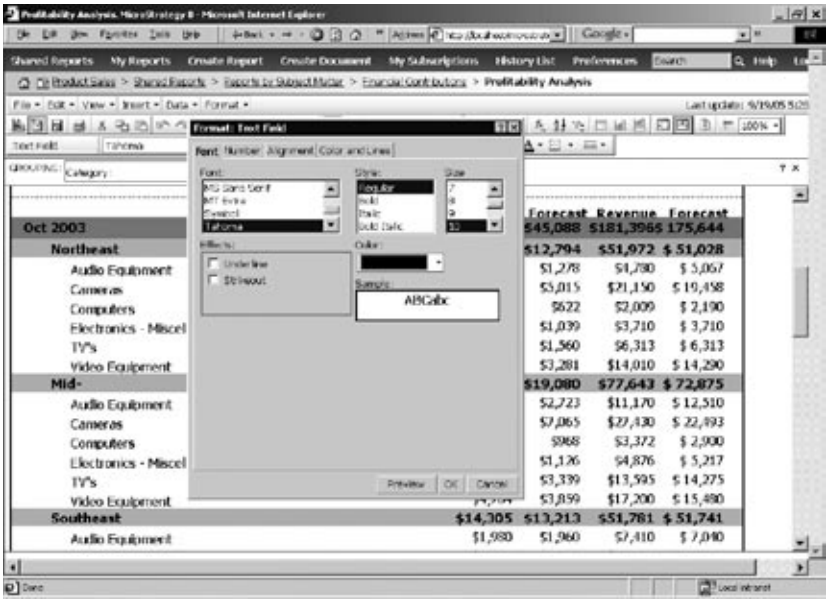


Figure 14.15 Data formatting – standard Excel-style formatting dialogs and functionality.

- Intuitive and Comprehensive Report Manipulations
  MicroStrategy users can perform several intuitive actions when viewing reports

  - One-click icons in toolbars with tool tips
  - Drop down menu option
  - Context sensitive right mouse click actions

These actions are presented in the graphic below, and include OLAP, formatting, export, and collaborative actions.

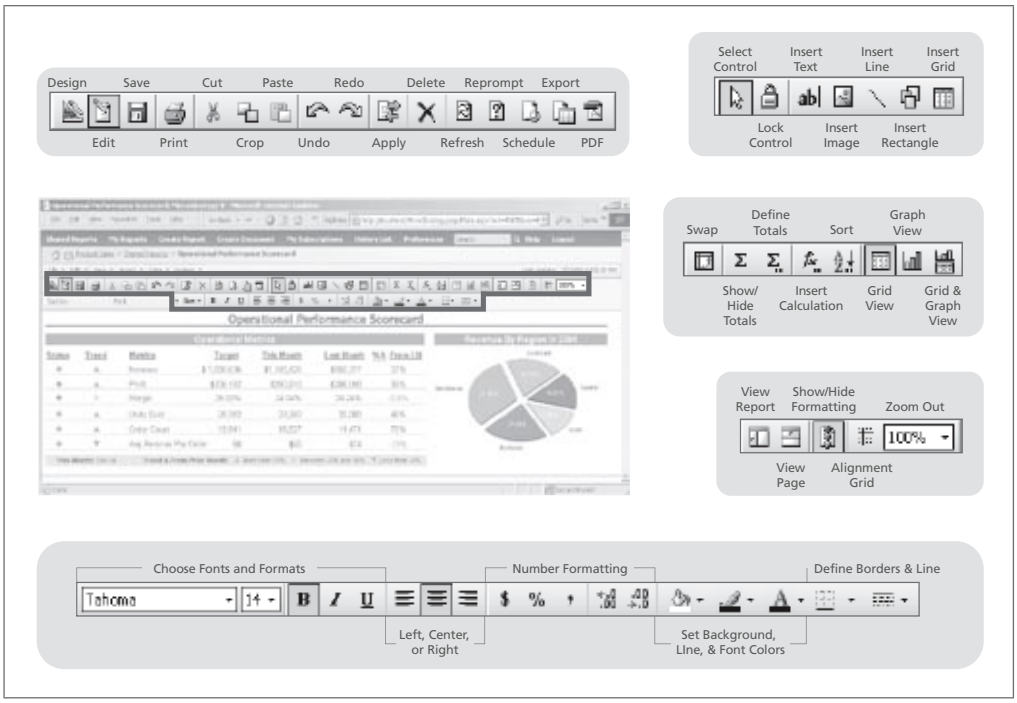


Figure 14.16 MicroStrategy Web contains a large number of report manipulation options.

- Extensive Report Export and Collaboration Options
  Once a report has been created, viewed, analyzed, modified, or manipulated, a user has several options to save, share or distribute the information. The report actions supported by MicroStrategy enable enterprises to deploy analytic applications that promote collaboration, and increase productivity. These report actions, which are available to both Web and Windows users, include the ability to:

  - Print. Print-perfect reports can be printed to any network printer. Users can dynamically change the page layout, apply fit-to-page functionality, and customize headers and footers for any report.
  - Email. Reports can be sent directly to other users to view and analyze with all the formatting of the original report in a variety of output formats:

    - HTML report image
    - PDF attachment
    - Excel attachment
    - Document with any combination of the above

  - Subscribe. Receive reports on a regular schedule or due to a triggered event. Reports can be delivered via email, to a printer or to a file server.
  - Save to personal folder. A report can be saved to a user's personal folder after data manipulation and formatting.
  - Save and publish a shared version. Reports can be saved in public folders for use by other business users. All security settings will be applied when other users run these reports.
  - Export. The report information can be exported, with or without formatting, to any of the following:

    - HTML
    - Microsoft Excel
    - Microsoft Word

- Microsoft Access
- PDF file
- Text file

- Schedule. Report caches can be refreshed from the data warehouse at off-peak hours, reducing the frequency of data source access via SQL or MDX.
- Create a datamart table. Automatically save the results from any report to a relational database table. This allows data-level integration with third-party applications like data mining or marketing automation systems.
- Write back to a database. The write back capabilities of the MicroStrategy BI platform allow enterprises to integrate analytics into their business processes and operational systems. Closed-loop analysis capabilities enable more accurate, timely, data-driven decisions by all business users.

14.6 OLAP Analysis

OLAP analysis refers to applications that require users to surf data, and make analytical deductions based on investigative reasoning. These applications require users to be able to access large amounts of data in an intuitive and rapid manner.

Unique MicroStrategy Strengths for Advanced and Predictive Applications

- Wizards and Workflows for Guided Reporting and Analysis
  Report wizards and navigation workflows in the MicroStrategy BI platform enhance the ease-of-use by guiding users through report creation, and through predefined drilling analysis. Users also have the option of building their own custom wizards and workflows. The types of wizards and workflows available are:

  - Intelligent Cubes
    MicroStrategy OLAP Services enables administrators and users to create sandboxes of analyses consisting of related characteristics, hierarchies, metrics, and attributes. These are available to users as "Intelligent Cubes" which are cached on the Intelligence Server in a multi-dimensional format. Designers can create a number of these Intelligent Cubes for various areas of analysis such as Customer Analysis or Financial Analysis, and make these available to end users over the Web. Users can then create their own analyses and reports based on these cubes by dragging and dropping the attributes and metrics of interest on to the report. This option allows users to create analyses in a tightly defined area, and also gives high performance due to the pre-cached nature of the Intelligent Cubes.

  - Report Builder Wizard
    This multi-step wizard guides a user through the steps required to build a report over the Web. Reports built using the report builder wizard are fully interactive, and can be saved, manipulated, shared, and distributed. A report builder wizard can contain any combination of attributes and metrics, or be directed to build a specific type of report. A BI application can have any number of report builder wizards. For instance, a financial reporting application may have both a profit and loss report builder, and a balance sheet report builder. The profit and loss report builder will limit the user to only income and expense information, while the balance sheet report builder will only access asset, liability, and capital information.

  - Ad Hoc Report Wizard
    The ad hoc report wizard allows a user to create and run an ad hoc report over the Web interface by combining an existing report template with an existing report filter. By allowing users to mix-and-match templates and filters from different reports, the ad hoc wizard greatly increases the reporting permutations

available to the end user without needing a new report for each permutation. Reports run using the ad hoc report wizard are fully interactive, and can be saved, manipulated, shared, and distributed. For example – a list of filters may include top decile of customers by revenue, and an available template may include regions and profit. Combining these together results in a report that shows the regional profit for the top decile of customers.

- Customized Drill Paths
  The MicroStrategy BI platform provides a drill map editor to define specific drill paths with which to navigate through data in the report. The drill path specifies the attributes, hierarchies, and templates that users can drill to from anywhere on the report for further analysis. This functionality provides users with an entire path of analysis within a given report. The user also has the option of saving a report during any point of analysis as a fully interactive report.

- Data Manipulation for OLAP Analysis
  The ability to drill, pivot, and slice-and-dice the information directly from a report is referred to as data manipulation. Data manipulation is central to any investigative business intelligence application since it allows the user to delve deep into a report to identify and discover important trends and patterns in the data. The MicroStrategy BI platform provides an extensive set of data manipulation options to both Web and Windows users through easily navigable interfaces. Using simple drag-and-drop, toolbar, and right-click mechanics, users are able to maximize the value of information from any given report. Report data can be manipulated, saved as a new report, and shared with other users.

  In many cases, the data needed for these actions are contained in the original report cache, and no new SQL or MDX statements are executed in the data source. If any objects are requested that are not in the original report, MicroStrategy Intelligence Server will automatically retrieve that information from the data warehouse, and create a new report cache. The following data manipulation functionality is available to both Web and Windows users. Microsoft Office users make use of the functionality built into Excel, PowerPoint, and Word.

  - Sorting
    Specify the sequence to sort one or more attributes or metrics directly on a report. The sort order, once defined, is preserved through any subsequent report manipulation steps. MicroStrategy supports single- and multi-key sorting on both columns and rows for complex reports with many attributes and metrics. For example on a customer sales report, users may first want to sort on customer region, and then order the customers by revenue within those regions. This is an example of a multi-key sort.

  - Pivoting
    Re-arrange the order and placement of attributes and metrics on both tabular and cross-tabular reports. Users can pivot their data between rows, columns, and pages using either drag-and-drop or single click actions. Pivoted reports can be saved in their new formats by the end user. For example, a report may have sales information with regions and years in columns. Users can switch or pivot the years to rows for a cleaner layout.

  - Page-by
    Create subsets of a report grouped by specific data elements. The page-by functionality is useful when viewing reports with many attributes and metrics. For example, a user can page-by region and month on a report to view specific information for a particular region and month. The user is then able to switch between different region and month values to change the focus of the report. MicroStrategy does not limit the number of attributes or combination of attributes that can be used with page-by to create the desired subsets of data.

- Drill Down

  Iteratively increase the level of detail of a summary report. MicroStrategy makes it easy for end users to view granular level information contained in the data of a report. For example, a user can drill down from a report at the quarter level to month, week or date in a single step.

- Drill Anywhere

  Perform train-of-thought analysis on any report. The drill anywhere functionality does not limit the drill path to a specific dimensions in a given report. For example, a user may be viewing a sales report at the regional level. The drill anywhere functionality will allow the user to drill from a given region to the supplier dimension to view vendor performance in each or all regions. Drill anywhere is a powerful feature that enables true 360-degree analysis of the information in a report down to the most granular level of data.

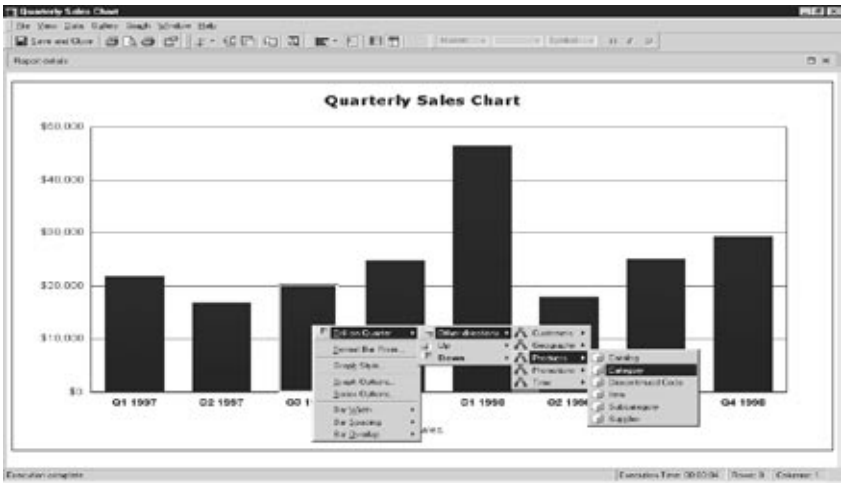


Figure 14.17 Data manipulation – drill anywhere functionality on graph and grid reports for train-of-thought analysis.

- Total/Subtotal

  Add totals and subtotals to metrics on a report. Totals and subtotals are supported for all types of metric calculations, including averages and percentages, and can be calculated on both rows and columns. In the case of averages or percentages, the total is calculated using the formula of the metric, rather than simply summing up the rows and columns. When totaling, users can choose from a variety of functions to calculate the subtotals and grand totals for the original metric calculations. New subtotal functions can also be added to the BI application. The built-in subtotal functions are:

  - Average
  - Count
  - Geometric Mean
  - Maximum
  - Median
  - Minimum
  - Mode
  - Product
  - Standard deviation
  - Sum
  - Variance

- Derived Metrics
  Create new metrics on the fly using existing metrics in the report. Functions available to users to build these on-the-fly metrics include common arithmetic operations and other functions to calculate moving averages, variances, standard deviations, and ranking operators. This functionality makes it possible for users to add a percent-to-total metric or a forecasting metric to a report. The new metrics are saved as part of the report definition for future use.

- One Click Metrics
  Metrics, such as ranking and percent to totals, as well as time series analysis including metrics like last years revenue, year to date revenue, can be quickly added to any report through a simple right click menu action.

- Quick Filtering of the Data (View Filters)
  Large reports and analyses often contain many rows of information with many measures to analyze. View Filters allow users to quickly narrow down large sets of information based on criteria specified by the user. For example, users looking at a report with sales information for all regions may only want to see north region information where revenue is greater than forecast by 20%. Users can quickly create a combination of conditions to get this information

- Lock Row and Column Headings
  For large reports, users can lock the row and column headers so that they don't lose context of the information as they scroll vertically or horizontally.

- Switch to Grid, Graph or Grid and Graph Views
  Users can quickly switch views between grid and graph views, or display both grid and graph information for a report at the same time.

- Fit to Contents or to Window
  Users can choose how they would like report data to be presented on the screen. Fit to Contents shrinks the columns to the standard width, whereas Fit to Window maximizes the column width to fill the screen.

• Sophisticated Parameter-driven Reports
  These reports accept user input at runtime to generate a customized view of a report. The MicroStrategy BI platform contains a sophisticated prompting engine that allows the selection of filtering criteria using attributes, such as a time period or a geographical attribute, and metrics, such as top 10% or bottom 10 by revenue. Report content becomes completely dynamic with object prompts. Users can choose the specific attributes and metrics from a list that are available on a report. This results in controlled, guided ad hoc analysis that can be deployed to all end users.

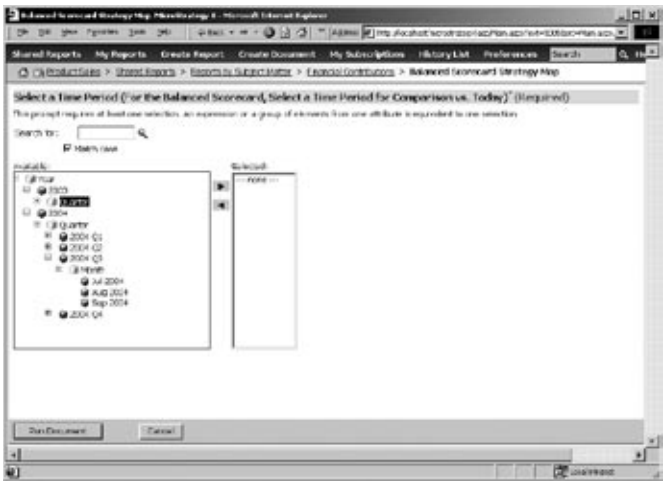


Figure 14.18 Hierarchy Prompt allowing users to select from various levels in a hierarchy.

MicroStrategy supports more types of user prompts than any other BI platform. User prompts allow the users to input or make selections at runtime to alter the contents of the report. Simple prompts may allow users to pick from a list of years to filter down time criteria for a report; more sophisticated prompts may allow users to select the level of a hierarchy and the metrics they would like to see on the report. MicroStrategy supports the following exhaustive set of prompts.

- Hierarchy Prompts allow users to choose elements from a hierarchy in a tree
- List Prompts include Shopping carts, Radio Buttons, Check Boxes, Multi-select Boxes formats
- Object Prompts allow users to choose which objects to place on a report at runtime
- Value Prompts allow users to specify values or thresholds for specific reports
- Level Prompts allow users to choose dimensionality for the metrics on a report

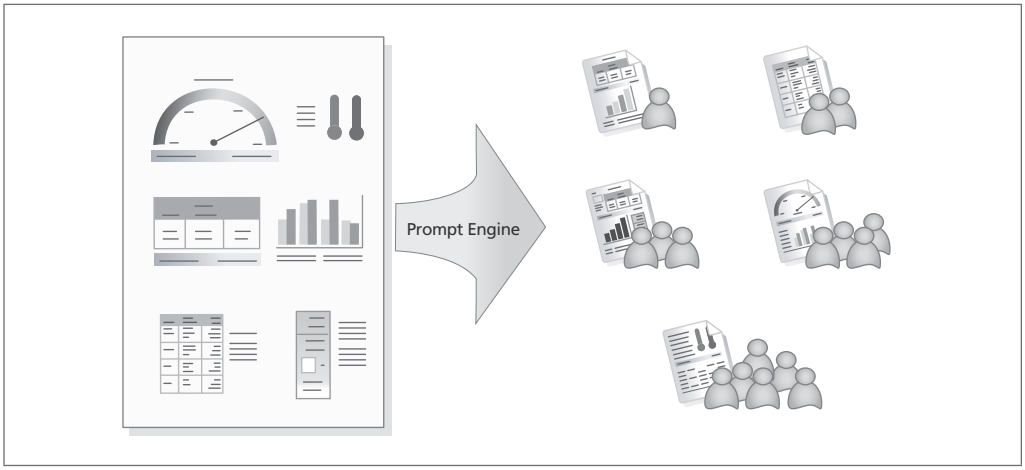


Figure 14.19 MicroStrategy allows the creation of a few reports that are personalized automatically at run-time across thousands of users using prompts.

- Seamless integration of Analytics into Scorecards

  Through years of focused design effort, and the development of proprietary algorithms and patents, MicroStrategy's highly refined relational OLAP engine meets the most demanding BI requirements.

MicroStrategy Web makes this same powerful analytic capability available directly from every report. It delivers analytic ability to business users in two ways:

- OLAP analysis can be performed directly on tables embedded within scorecards, dashboards, documents, and reports.

- Drilling is possible from an enterprise report or scorecard to a dedicated view that is optimized for conducting detailed analysis from the same unified Web interface.

In both cases, the integration of reporting with analysis is automatic and entirely seamless, enabling self-service reporting, analysis, and monitoring by all business users. Business users can answer their own follow-on questions as they occur, without need of further IT department support.

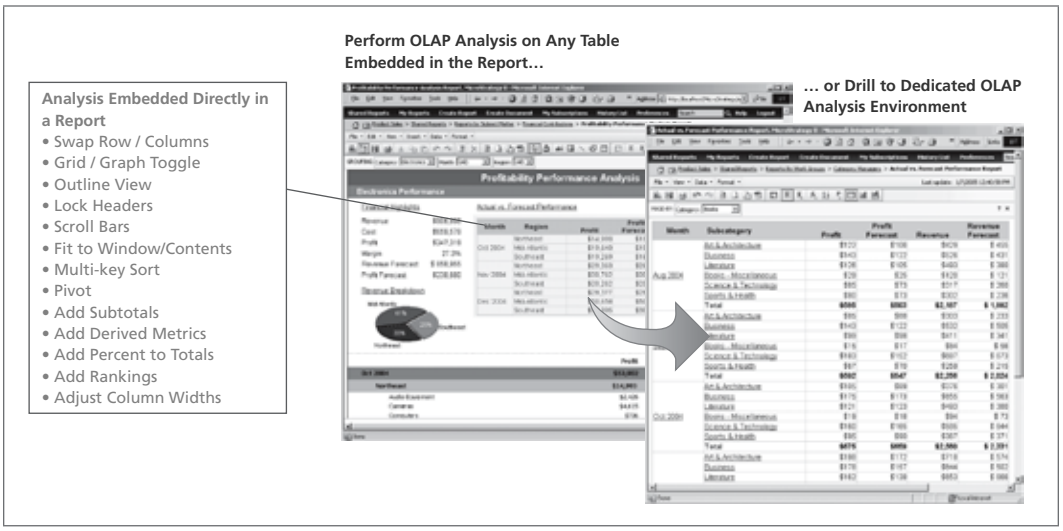


Figure 14.20 Further analysis is possible on any report, scorecard or dashboard.

14.7 Advanced and Predictive Analysis

Unique MicroStrategy Strengths for Advanced and Predictive Applications

Ad hoc reporting capabilities reduce development and administrative burdens by transferring report creation to the end user. The MicroStrategy BI platform provides extensive ad hoc capabilities for both Web and Windows users, who can build reports, modify reports, save reports, and dynamically change the attributes and metrics on a report.

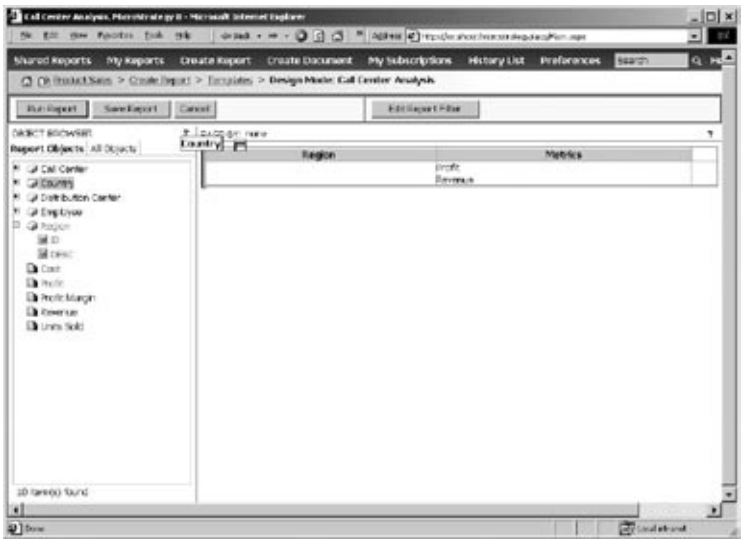


Figure 14.21 Ad hoc report creation over the Web with MicroStrategy.

Set analysis reports use sets of attributes that can interact with each other to provide powerful analysis. Users define complex criteria for finely segmenting customers by merging sets with each other using standard set operators. MicroStrategy provides a powerful and fully integrated set of set analysis capabilities in the Web and Windows interfaces.

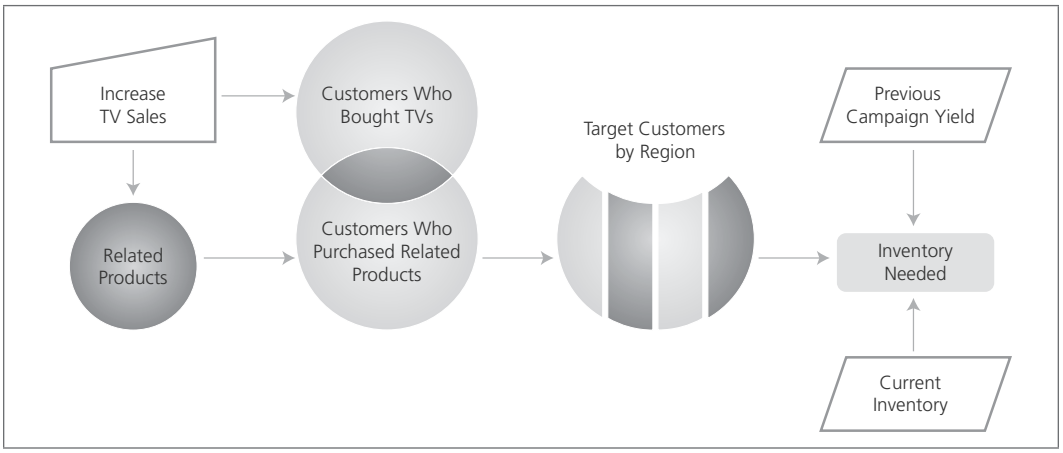


Figure 14.22 Set Analysis with MicroStrategy provides very powerful analysis capabilities.

What-if analysis reports allow users to identify outcomes based on different scenarios. Metrics in MicroStrategy use numeric prompts combined with aggregated information from the data warehouse to provide what-if calculations. For example, a revenue forecast can be generated by applying a prompted percentage to the actual revenue. What-if reports can also be run directly in Excel, which provides its own what-if environment.

Advanced Analysis and data mining reports deliver complex calculations and predictive metrics to all end users, enabling more accurate data-driven decision making. Metrics using built-in statistical, mathematical, financial, and data mining algorithms add analytic richness to any BI application. These reports and analyses are available in all MicroStrategy BI interfaces without any additional programming or knowledge of how the functions and algorithms work.

14.8 Alerts and Proactive Notifications

Collaboration applications integrate corporate-wide information so that all users, both within and outside an enterprise, have the same version of the truth at all times. Collaborative analytic applications must therefore account for varying levels of user expertise, multiple interface and interactivity requirements, and the need for on-demand and proactive information delivery. The MicroStrategy platform provides users with an extensive array of interface choices and information delivery options. MicroStrategy's sophisticated reporting capabilities also allow end users to perform complex collaborative tasks, such as analyzing information, exporting data to other third-party applications, writing back to the database, and sending email alerts. All of this can be done through a single user interface.

Collaborative applications can operate in 2 modes:

1. Administrator-led Push Applications
    - Information can be distributed to large user populations via print, email or file servers, with each user recipient automatically getting personalized information based on their security profiles. Users can receive information either on a time or event-driven schedule

2. User-led Proactive Applications in the MicroStrategy Web Interface
    - Subscribe to reports to be sent via email, print or file server based on a schedule or event
    - Send reports to other users via email, print or fax

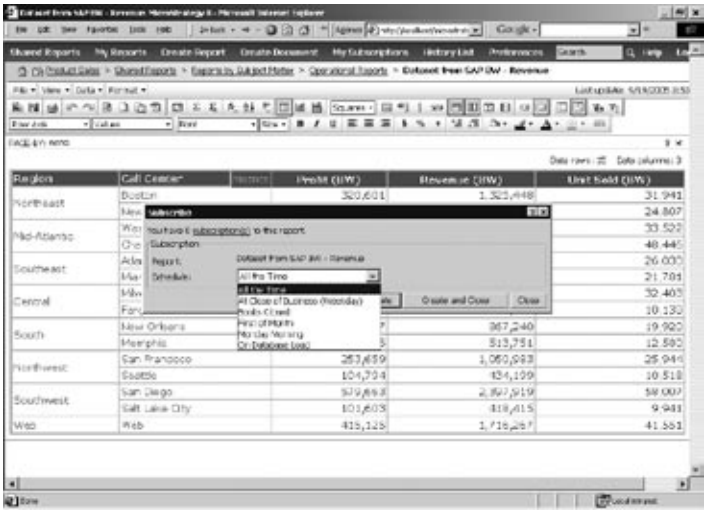


Figure 14.23 MicroStrategy Web allows the subscription of reports using predefined schedules.

14.9 User Functionality by User Role

In addition to growth in the number and types of business intelligence applications, as detailed above, the number and types of users of these applications has expanded. Architectures for enterprise business intelligence need to be able to accommodate the needs and functionality of all types of users, from analysts and power users to information consumers, in a scalable and manageable manner that does not require extensive customization or coding. Furthermore, reports and analyses developed by any user must be shareable as they may be needed by other users.

The Business Intelligence User Spectrum

The main types of users and their relative distribution amongst the user population in a typical enterprise are shown in the diagram below:

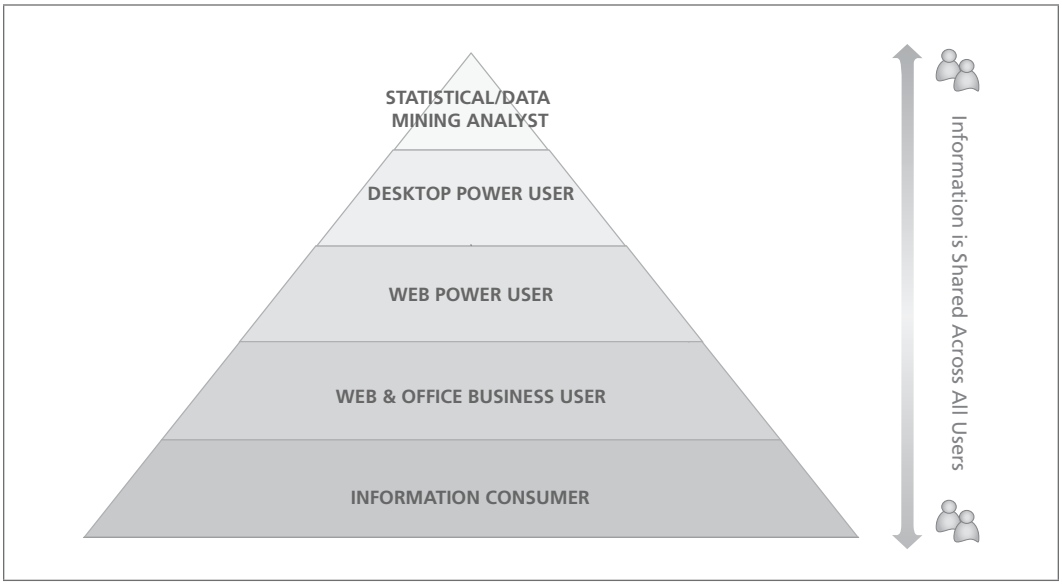


Figure 14.24 The number of BI users in an enterprise is defined by the functionality they use.

1. Information Consumer
   This user receives static information either via the Web or via scheduled delivery to email or other devices. The information is well-formatted, and designed to be consumed quickly and easily, with little interactivity.

2. Web and Office Business User
   This user uses a Web interface to run reports and documents. The reports allow some OLAP actions, such as pivoting, sorting, and drilling, and can be run from or exported to Microsoft Office applications. The ability to reformat and edit reports in a WYSIWYG manner is often also required.

3. Web Power User
   This user runs both standard and ad hoc reports and documents from MicroStrategy Web, and performs advanced OLAP actions such as drilling anywhere to transaction-level data, adding time-series analysis metrics to reports, and saving documents and reports that other users can consume.

4. Desktop Power User
   This user needs advanced functionality that defines the metadata objects used to build documents, reports, and analyses. This includes creating sophisticated metrics, arranging data into custom groups to satisfy particular groupings, and defining filters for set analysis.

5. Statistical/Data Mining Analyst
   This user develops advanced statistical and data mining models, using neural networks, regression, and clustering algorithms, to create scoring and confidence predictive metrics. These predictive metrics can be incorporated into any document or report, and are available to all users.

MicroStrategy provides the functionality to satisfy all reporting requirements for all BI users. More importantly, BI administrators retain a very precise degree of control over exactly what functionality is given to a specific user, a group of users or a user role. Users from two different groups can access the same report through any interface with the exact functionality that is appropriate for each user. For example, a user from the "Reporting" group might have more limited interactive capabilities than a user from the "Analyst" group. This protects the BI system, the database, and the business users themselves from unnecessary complexity, while still providing the full range of easy-to-use interactivity to those users who benefit from it. Even within MicroStrategy's end-user products, BI administrators can further adjust and personalize an individual user's, security role's or user group's access to MicroStrategy functionality.

User Functionality Governed by Privileges

The MicroStrategy Architecture allows all of MicroStrategy's interfaces to be customized to the target user population (from novice to expert) through extensive use of privilege controls. Users or groups of users can be assigned varying levels of functionality such as pivot, print, and send now on a per application or project basis. In addition, users can be assigned security profiles to ensure that they can only see the data and objects in folders that they can access. Thus, without writing a single line of code, administrators can configure an application and user interfaces to suit all user populations.



Figure 14.25 Business users are able to view all types of reports and scorecards, and also personalize reports, print, drill, sort, choose grid or graph format, schedule or immediately send reports via email or to a file server or a printer.

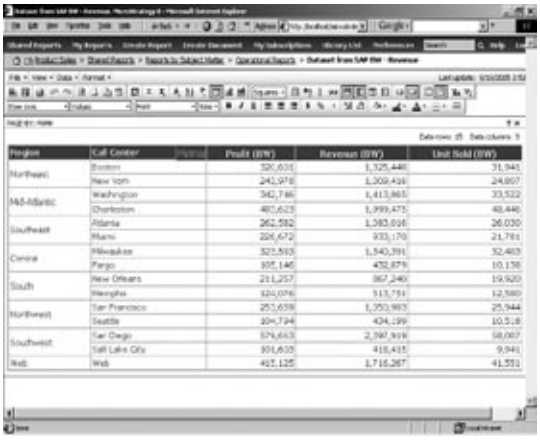


Figure 14.26 Analysts receive all the functionality of business users plus the ability to drill anywhere, add totals, pivot information, add and remove fields, create derived metrics, and create reports or ad hoc queries.



Figure 14.27 Power users get all the functionality of analysts plus the ability to adjust the format of scorecards, dashboards, and operational reports in design or WYSIWYG view mode. Power Users have advanced formatting capabilities as well as the ability to perform calculations across multiple data sources.

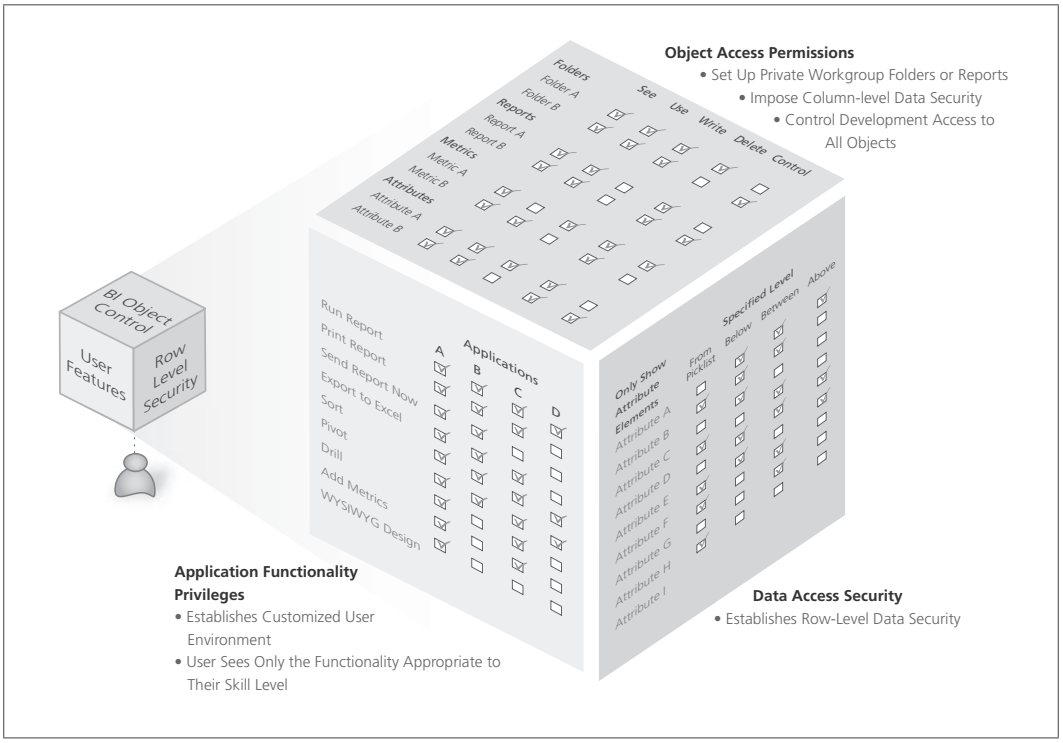


Figure 14.28 User functionality control through Privileges, ACL's and Security filters.

Personalized reports are generated from a single report definition so that report contents are restricted to information that an end user is allowed to see. For example, the same Sales Report will show only West Region details to the Western Regional Manager, and show only East Region details to the Eastern Regional Manager. This personalization is applied in Web, Windows, and Microsoft Office interfaces, and also when users subscribe to a scheduled or alert report, ensuring both data security and delivery of only relevant information to the user.

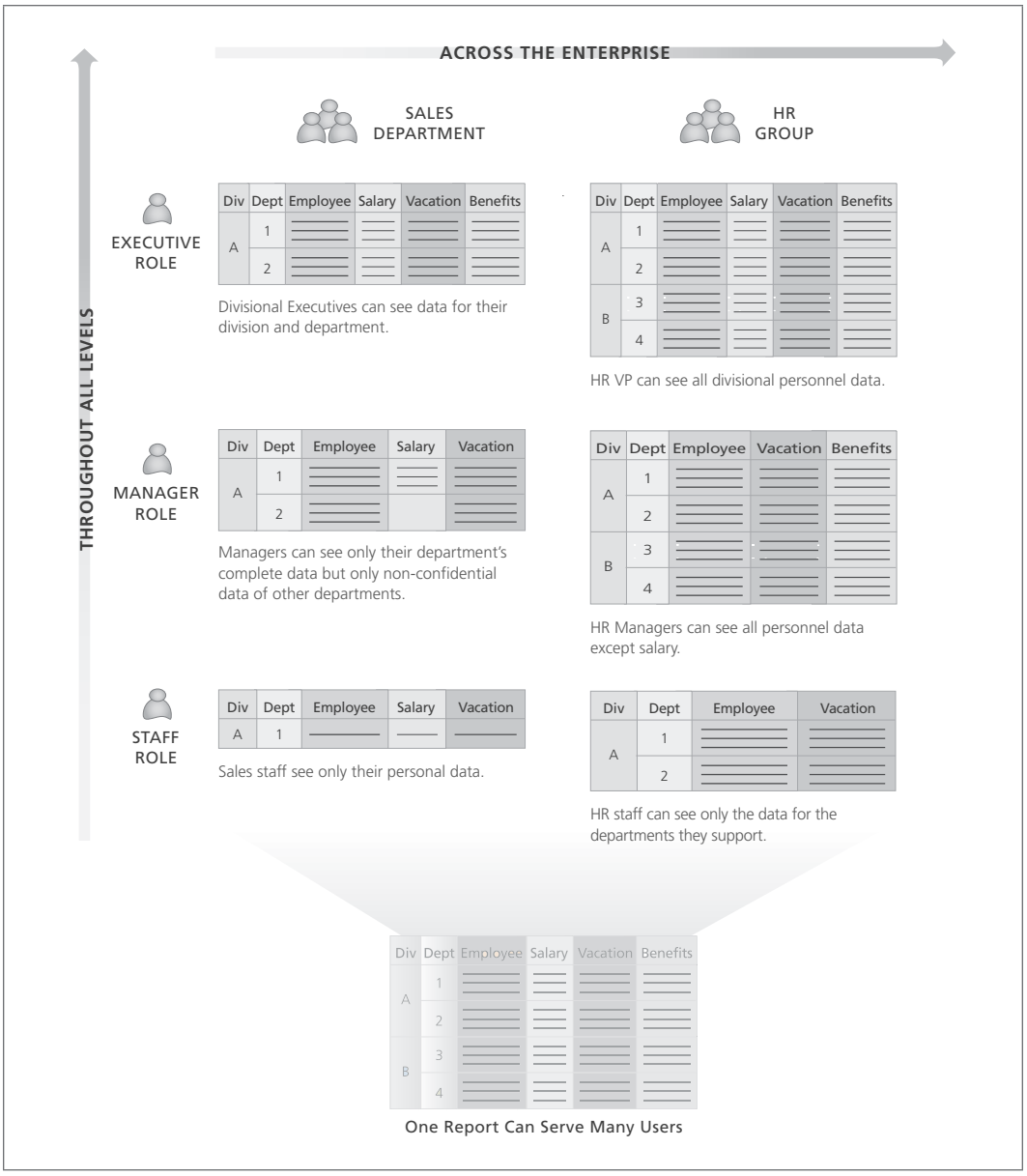


Figure 14.29 Reports are automatically personalized to each user's security profile.

14.10 Customizing MicroStrategy Functionality for the End User

Although the MicroStrategy BI platform provides a comprehensive set of analytical features and functionality through a single interface, enterprises can add their own custom functionality. MicroStrategy's open architecture and APIs, which are documented extensively in the MicroStrategy SDK, make it especially suitable for developing custom functionality or integrating with other applications. The SDK describes how to integrate with enterprise portals, external security systems, and Web services applications. The goal of the MicroStrategy platform architecture is to enable enterprises to fully leverage their entire information assets across the enterprise.

14.11 Summary

The MicroStrategy BI platform has been architected from the ground up to provide an intuitive, consistent, and easy-to-use experience for users. The hallmarks of the MicroStrategy BI platform for user experience are:

- Consistent and intuitive user experience across all interfaces including MicroStrategy Web, MicroStrategy Desktop, and MicroStrategy Office
- Support for all 5 styles of BI and seamless user navigation between any of these styles of BI within the same interface
- Powerful Distribution and Subscription interfaces for push reporting
- Interfaces governed by privilege settings to easily configure for all users ranging from novice to power users
- Comprehensive report, OLAP, and formatting options from all interfaces to allow users to create, edit, and modify reports

15

Developer Experience

A key component to building successful BI applications is having a robust set of developer tools to enable developers of all levels to create reports and other metadata objects rapidly, intuitively, and in a scalable, manageable manner.

15.1 Multiple Interfaces for BI Development

The MicroStrategy BI platform offers several interfaces and tools for the BI developer all tied together with a secure and object-oriented metadata. The following products and interfaces are available to create and maintain BI applications:

1. MicroStrategy Architect – Used to map data models (in relational databases, SAP, and other cube databases) to business models and to create attributes, facts, and hierarchies
2. MicroStrategy Desktop Designer – Windows-based design interface developing filters, metrics, reports, consolidations, and custom groups
3. MicroStrategy Web Professional – Thin-client browser interface that allows users to create reports, dashboards, and scorecards over the Web using an intuitive drag and drop interface
4. MicroStrategy Narrowcast Server – Batch and alert reporting product that allows developers to create exception and alert-based applications to deliver information via portals, emails, file servers, and network printers
5. MicroStrategy SDK – The MicroStrategy Software Developer Kit allows developers to customize the look and feel, and content of the MicroStrategy interfaces, and extend the functionality available within the platform

15.2 The Business Intelligence Application Spectrum

MicroStrategy Architect is a rapid BI development tool that maps the physical structure of a data source to a logical, object-oriented model of the business through an intuitive, graphical interface. Architect provides wizards and editors to define the business rules, providing a unified environment for creating and maintaining business intelligence applications. BI architects use Architect to:

- Map the physical database schema into a logical business model using wizards and editors.
- Model the application using business processes and terminology.
- Build complex hierarchies that reflect the relationship between business entities using one-to-one, one-to-many, and many-to-many relationships.

The business rules and schema abstraction defined using MicroStrategy Architect are stored in the central metadata repository, allowing MicroStrategy BI products to reuse the definitions, and build upon them. These are stored as schema abstraction objects in the metadata repository.

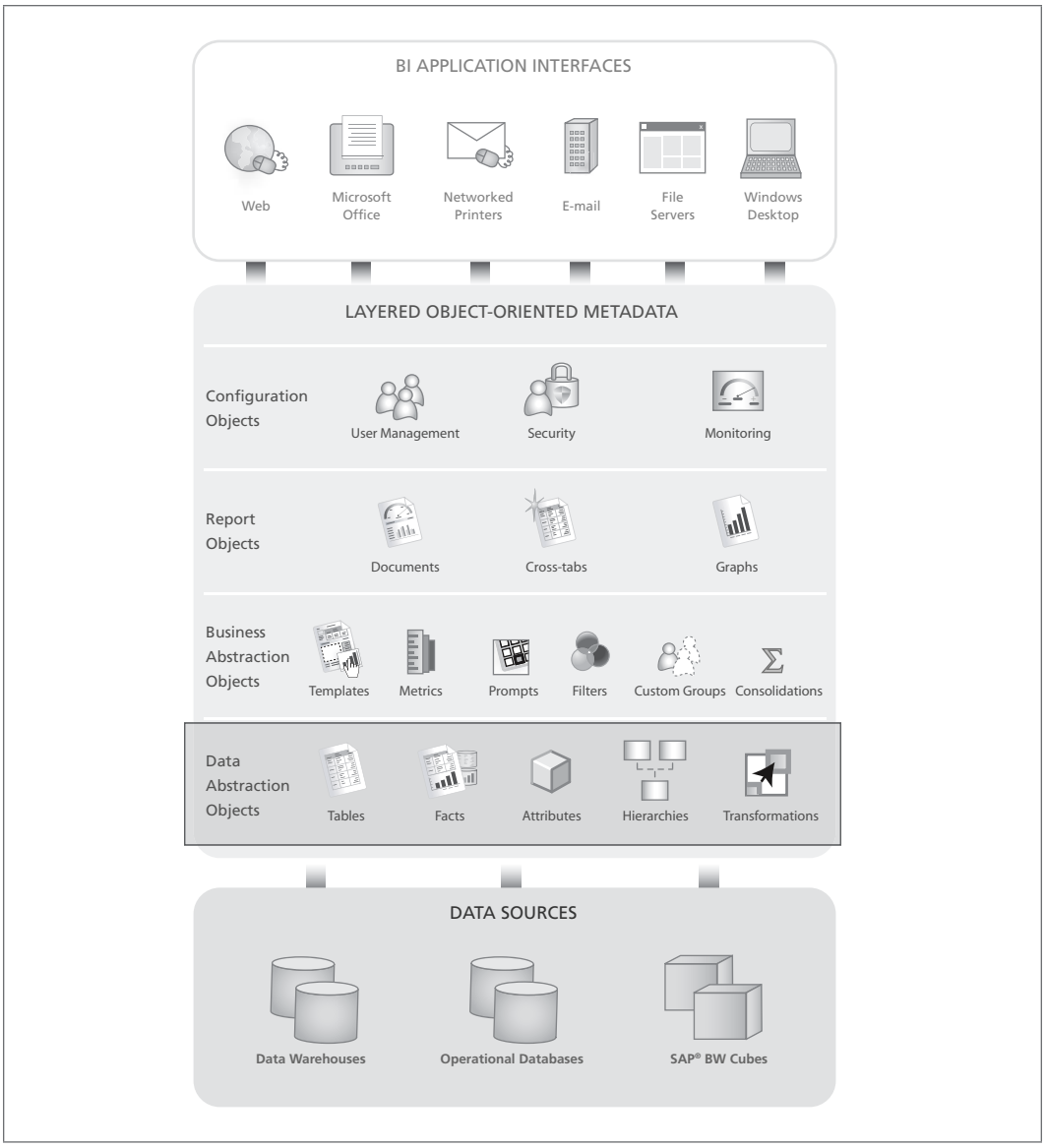


Figure 15.1 MicroStrategy Architect is used to create the data abstraction of the data sources used by the BI applications.

Object Creation in MicroStrategy Architect

Using MicroStrategy Architect, system designers can create data abstraction objects, which map data sources to the business model, and are used to create higher level objects such as reports and scorecards.

The following are the data abstraction objects that can be created using MicroStrategy Architect

- Attributes
  Attributes define descriptive information in the data, such as Customer Name, Product Category or Current Date. They are created in MicroStrategy by mapping business descriptions to the corresponding data elements in the data source. Developers can also specify the parent-child relationship between attributes as well as create expressions based on multiple data fields. These are discussed in further detail in later sections.

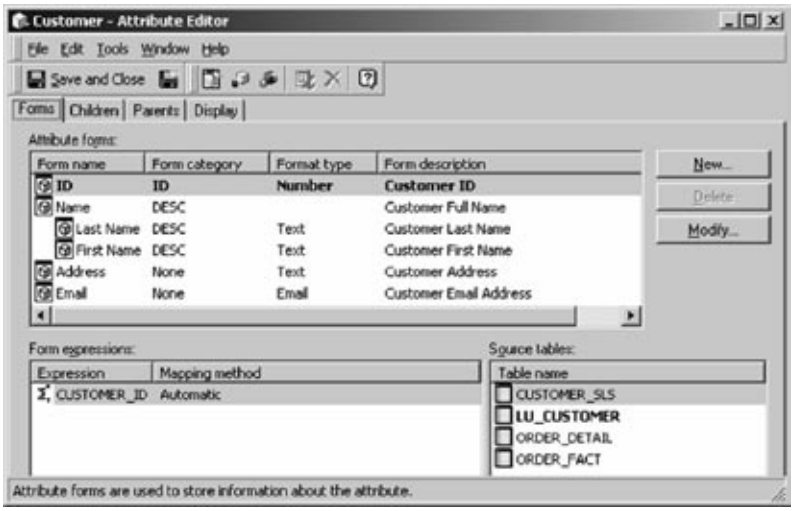


Figure 15.2 Attributes in Architect can be automatically mapped by column name.

- Facts

  Facts are data abstraction objects in MicroStrategy that are used to define measures such as Order Amount, Number of Customers or Current Inventory on Hand. These are usually based on numerical data sources. Similar to attributes, users can use sophisticated expressions to define complex facts from multiple data source inputs.

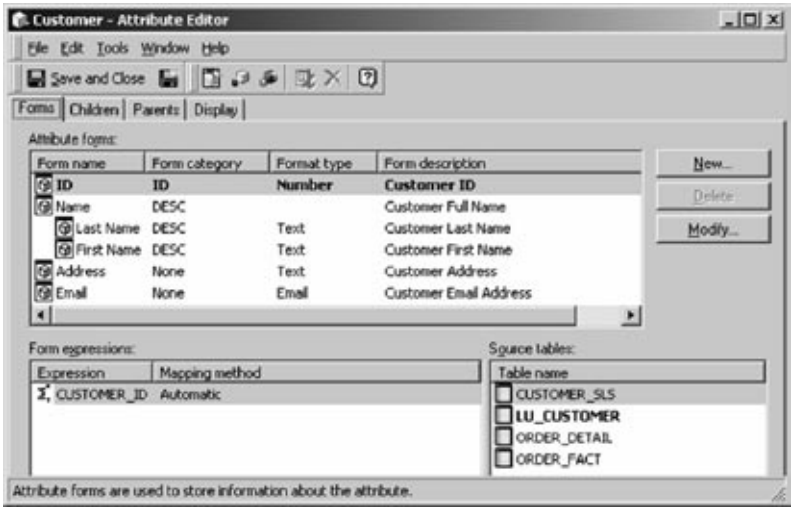


Figure 15.3 Facts are mapped to different database tables providing automatic aggregate awareness.

- Hierarchies

  Hierarchies are used in the MicroStrategy environment to logically group attribute elements into areas of commonality. Typical examples include the Time Hierarchy, Sales Organization Hierarchy or Product Hierarchy. There are two types of hierarchies in the MicroStrategy environment:

  i. System Hierarchy – This hierarchy is created automatically when parent-child relationships are defined between attributes. The system hierarchy mirrors the structure of the data in the data source
  ii. User Hierarchies – These hierarchies can be created by users independent of the data source structures. Attributes can be grouped together to form logically coherent groupings based on business models to simplify reporting for end users. There is no limit on the number and sophistication of user hierarchies.

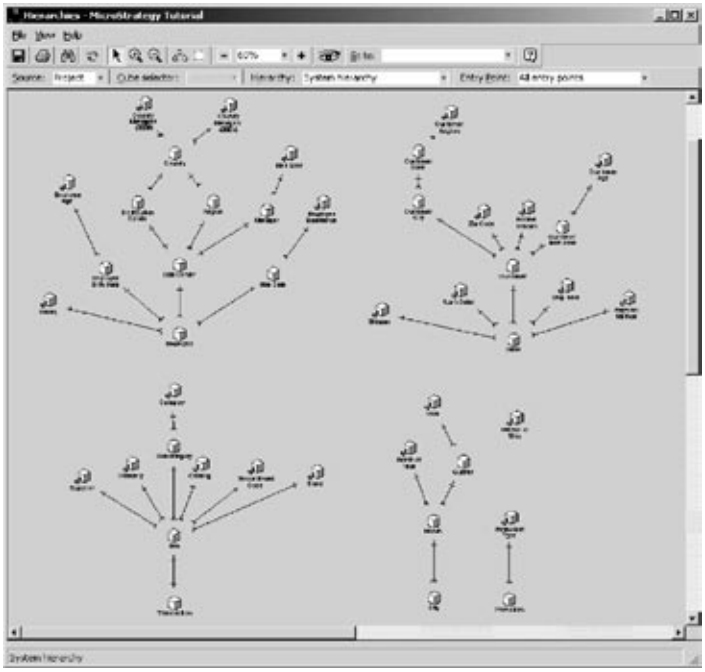


Figure 15.4 Hierarchies can be viewed graphically in Architect.

- Transformations

  Transformations are a powerful concept used to build sophisticated analytical applications. They are used to transform between sets of attributes. For example, architects can build time transformations such as "Last year's Revenue" or "Quarter to Date Revenue." These can then be applied to metrics by end users with a single click to get the relevant data transformed over time.



Figure 15.5 Transformations are mainly used for time-series analysis and have 1-to-1 or many-to-many mappings.

- Partitions

  When dealing with large data volumes, data is often partitioned in databases to improve performance. For example, instead of storing detailed sales order records for all months in a single table, DBA's often break this table down into a set of tables partitioned by month so that any one table contains only orders for a single

month. Most databases allow for partitions to be created, and the optimizer automatically routes the query to the correct table. MicroStrategy Architect supports this partitioning functionality natively. In addition, users can create logical partitions within MicroStrategy that perform the same functions described above without relying on the database to perform the query optimization. MicroStrategy's advanced multi-pass SQL engine creates automatic queries based on partition logic.

- Logical Tables
  Logical tables are powerful tools that allow the OLAP architect to create views of information without creating and maintaining these views in the database. Similar to database views, logical views are created by typing in any valid SQL statement against a database to create the view. Once this view is created, it can be used just like any other table within MicroStrategy. The unique advantages that this confers include:

  - Defining attributes and facts from expressions spanning multiple tables
  - Flattening non-standard recursive hierarchies
  - Reducing the number of database views
  - Performing complex analysis, such as as-is versus as-was for slowly changing dimensions.

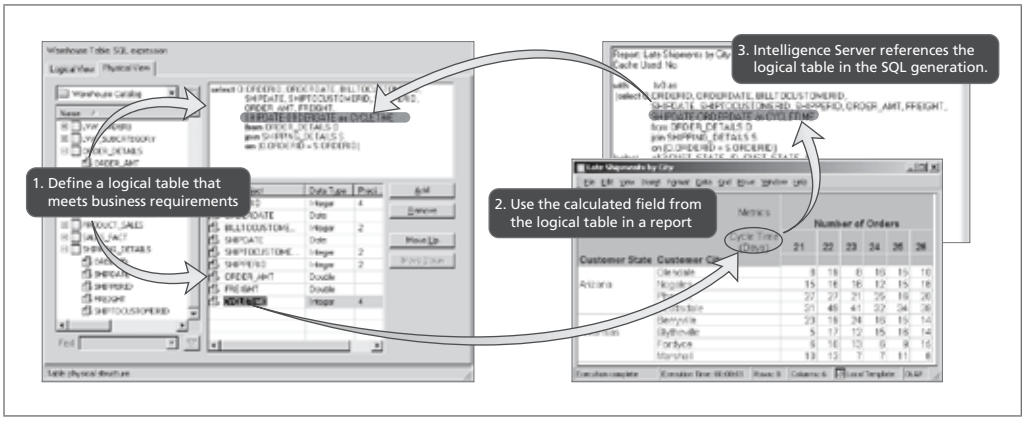


Figure 15.6 A logical table using freeform SQL embeds the SQL statement as a derived table in the dynamically generated report SQL.

Unique Technical Differentiators of MicroStrategy Architect

MicroStrategy Architect offers the most comprehensive and sophisticated business modeling tool available today in the BI industry. The following section discusses the unique strengths of MicroStrategy Architect for the OLAP architect.

15.3 Map Database Structures to Qualitative and Quantitative Business Terms

MicroStrategy Architect reads the system catalog from the data warehouse, operational datamarts, and databases, as well as cube databases such as SAP BW, and translates the definitions of these data source structures as logical tables and columns. The table and column abstraction objects are the basis for other data abstraction objects such as facts, attributes, hierarchies, and transformations.

This schema abstraction of the physical database tables defines a logical model of the business requirements rather than just the database structures. Therefore, if the underlying physical data model changes, BI architects can quickly and easily incorporate these changes without affecting the higher level objects that use the schema layer. This insulates the BI application from ongoing database modifications.

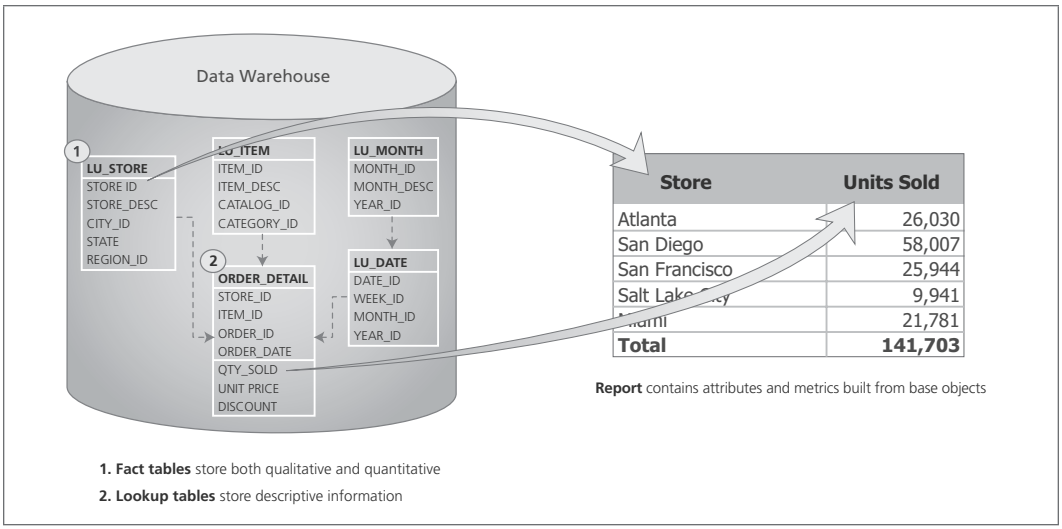


Figure 15.7 Architect allows BI architects to map data source columns to business terms using attributes and facts.

User Hierarchies Personalize Business-Focused Data Exploration and Investigative Analysis

BI architects model the relationship between attributes as hierarchies. The system hierarchy represents the true relationship between attributes necessary to generate the SQL when users run reports. The system hierarchy is automatically defined by Architect when attributes are created, and their physical database relationships to other attributes are specified. When users run reports or drill through data, Intelligence Server reads the system hierarchy in order to generate the appropriate SQL to satisfy the requests.

MicroStrategy Architect provides a unique business abstraction for groups of attributes in a single BI application called user hierarchies. These allow users to browse and drill through the data as required by business needs, as opposed to how the data is physically stored in the data source. User hierarchies can be created through graphical viewers and editors. The visual map displayed in these viewers allows developers to quickly identify and create relationships between data elements. Additionally, business users can graphically visualize the business model.

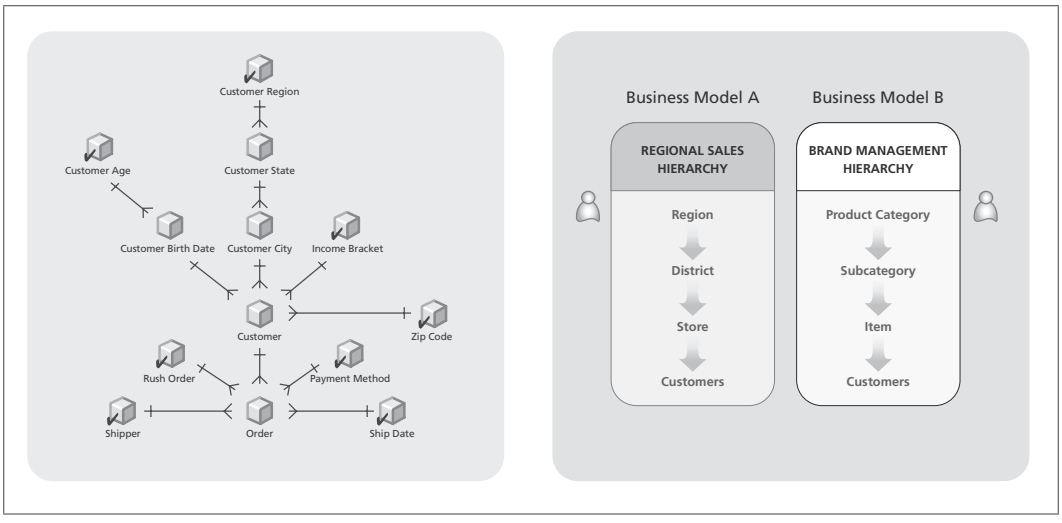


Figure 15.8 The system hierarchy maps the physical data structure. Many user hierarchies define logical business hierarchies that are often different to the way data is stored.

Support Any Data Schema

The MicroStrategy architecture supports a wide range of physical data schemas, including common ones such as star and snowflake schemas that have sparse aggregate tables, partitioned tables, and split fact tables. It can also accommodate direct access to operational data stored in third-normal form. Support for a broad range of relational and multi-dimensional data warehouse schema designs ensures that reporting applications easily integrate with an organization's existing data sources.

Leverage Existing Investment in Data Warehouses

Some of the common challenges faced by BI architects while modeling BI applications that leverage existing investments in a data warehouse include overcoming:

- Inconsistent naming of database fields that represent the same entity
- Entities stored across multiple database fields and tables
- Unavailability of data at levels expected by the business users

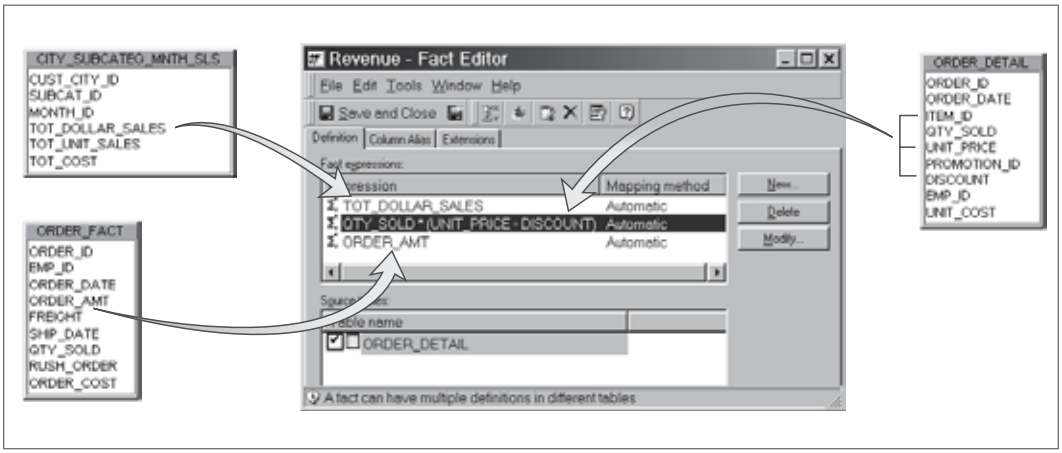


Figure 15.9 The same MicroStrategy object can have different definitions based on the source of the data.

BI architects can easily make available to users different representations of the same attribute by defining attribute forms. Non-textual descriptors, such as images, HTML tags, and URLs, are supported attribute form types that provide great visual appeal and report interactivity.

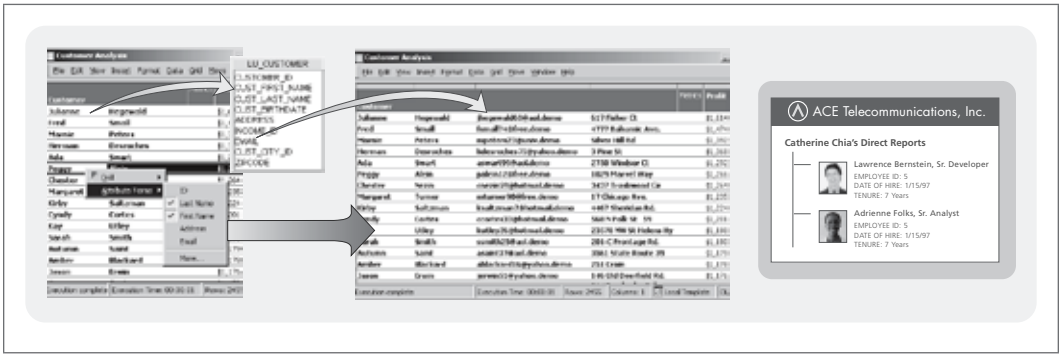


Figure 15.10 Increase visual appeal of reports by defining attribute forms that use dynamic images or URLs.

Define Multiple Attributes from the Same Database Field Using Attribute Roles

BI applications often need to define many attributes that use the same database table and column. For example, an airline may have a single lookup table that lists all the airport locations served by the airline. To report on timeliness or traffic volumes between airports, the reports need to display airports by the flight origination and destination. MicroStrategy Architect provides unique data modeling options that allow the BI architect to define multiple attributes based on the same lookup table without resorting to database views that may add to the database administration and maintenance burden.

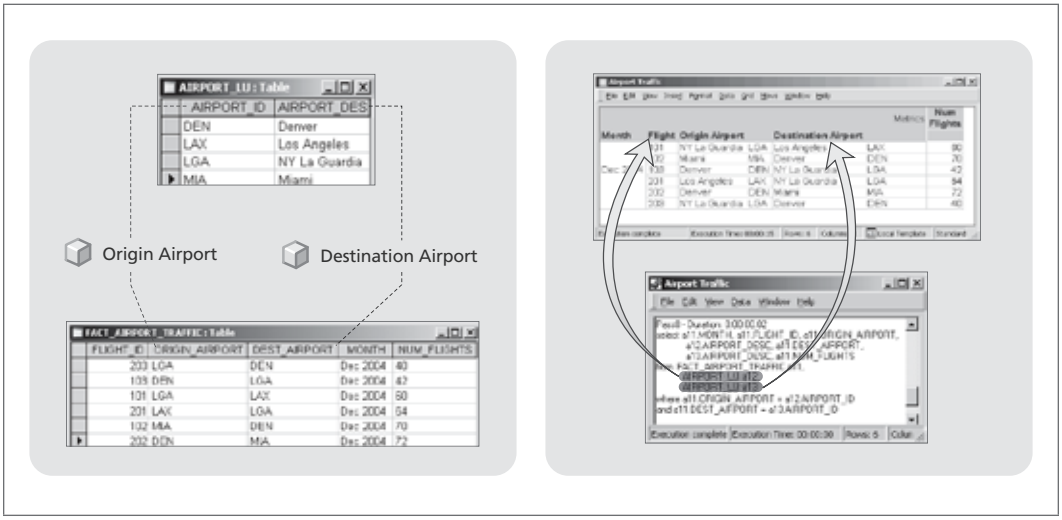


Figure 15.11 Fields in a single lookup table play multiple roles: An airport can either be the origin of a flight, or the destination of the flight as stored in the fact table.

15.4 MicroStrategy Desktop

MicroStrategy Desktop is a powerful development interface that includes integrated performance monitoring, enterprise reporting, ad hoc analysis and decision support workflow. Desktop provides the means to easily access and share critical corporate information from a variety of corporate data sources such as data warehouses, data marts, operational databases, and SAP BW, in order to better manage business processes.

With Desktop, report developers get a comprehensive tool to develop the most sophisticated BI applications:

- Build analysis objects using the metadata abstraction layer provided by the data abstraction objects and other analysis objects
- Design and publish reports and documents by combining analysis objects into report objects for powerful analysis

All the objects and business rules defined with Desktop are stored in the central metadata repository, providing a consistent definition and view across all MicroStrategy BI.

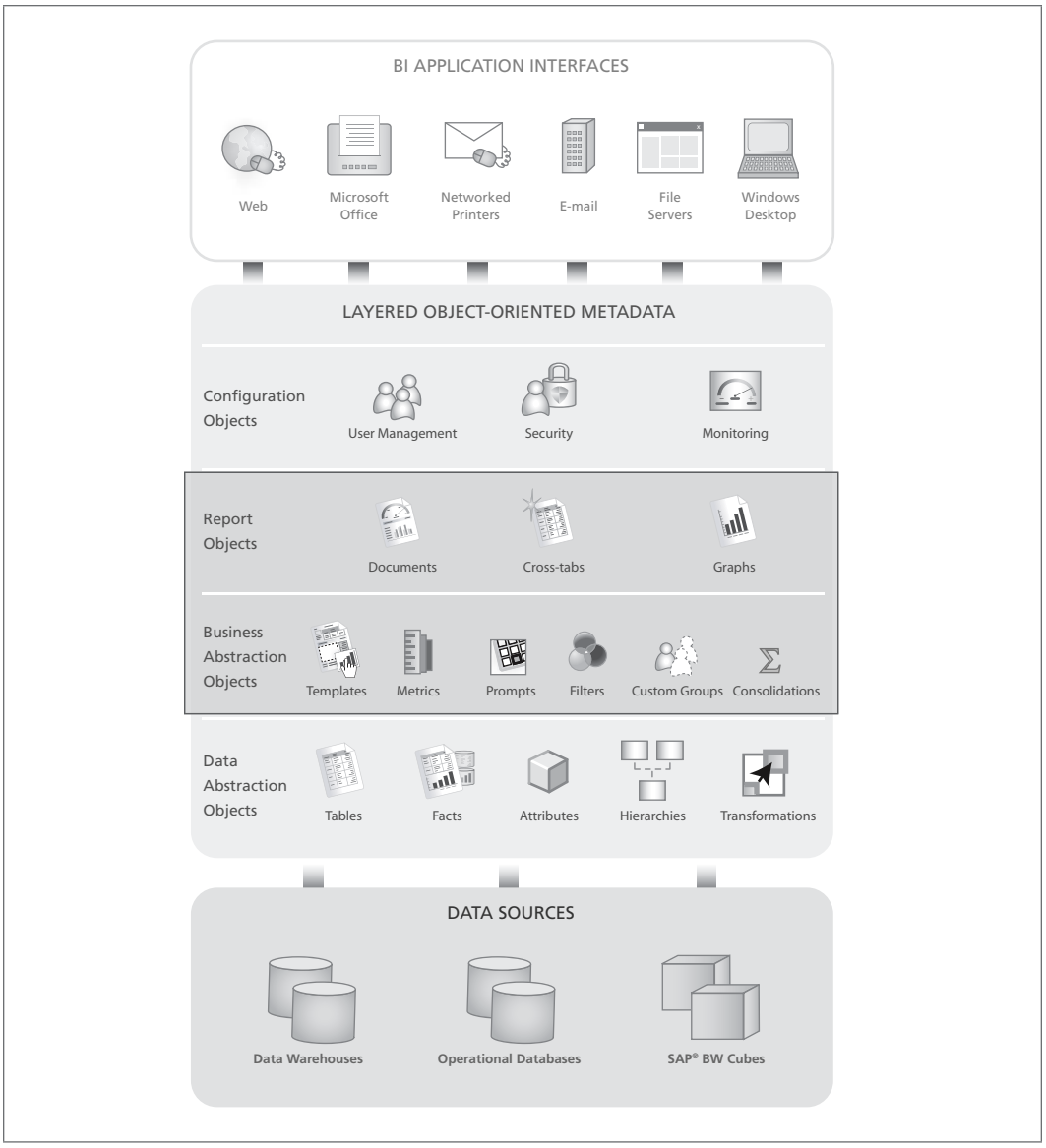


Figure 15.12 MicroStrategy Desktop allows organizations to design business abstraction and report objects used to create sophisticated BI applications. All of these are built on top of the data abstraction of the database schema created using MicroStrategy Architect.

Objects Created in MicroStrategy Desktop

BI developers use MicroStrategy Desktop to create two sets of objects:

1. Business Abstraction Objects
   Application objects are base objects that are building blocks for reports, scorecards, and dashboards. These include metrics, filters, templates, and custom groups. Application objects can be shared by multiple reports.

2. Report Objects
   Report objects are built using a combination of business abstraction objects. Grid reports, graph reports, scorecards, and dashboards are types of report objects.

15.5 Business Abstraction Objects

- Metrics

  Metrics define the analytical calculations to be performed against data in the data warehouse. A metric definition must contain a formula, which determines the data to be used and the calculations to be performed on the data. It can also contain additional properties that provide great flexibility in defining key performance indicators:

  - The level, or dimensionality, determines the level of calculation for the metric. For example, you can choose to calculate at the month level or the region level. A simple metric can only contain one level. By default, a metric is calculated at the report level, that is, at the lowest level attributes of the report in which the metric is placed. For example, a report with revenue and state will calculate revenue at the state level unless dimensionality to set to country, in which case revenue will be calculated at the country level even if it is not on the report.

  - Conditionality associates a filter to the metric calculation. This filter can be controlled to work in conjunction with the report filter. This is an optional component.

  - A transformation applies offset values, such as "four months ago," to the selected attributes. This is also an optional component.

  Metrics allow complex business-specific calculations to be presented on reports or documents. Report developers can use one or more functions from an analytic function library that includes aggregation, OLAP, statistical, financial, mathematical or ranking functions as well as logical operators.

  The object-oriented nature of the metadata allows metrics to reuse other schema and analysis objects, such as attributes, facts, metrics, filters, and transformations, to define new metrics. Using filters and transformations in the metric definition allows users to create sophisticated conditional and comparison calculations.



Figure 15.13 The metric editor in Desktop provides great flexibility to create any key performance indicator.

- Filters

  A filter specifies the conditions the data must meet to be included in report results. A filter is equivalent to the WHERE clause in a SQL statement. Developers can use the following options to create different types of filters:

  i. Attribute qualification – allows you to filter on an attribute's form, such as ID and description, or on the elements of the attribute.
  ii. Set qualification – allows you to create a set based on a metric or on the relationship between two attributes
  iii. Report qualification – uses an existing report as a filter
  iv. Filter qualification –uses an existing filter as a base to add further conditions to a report

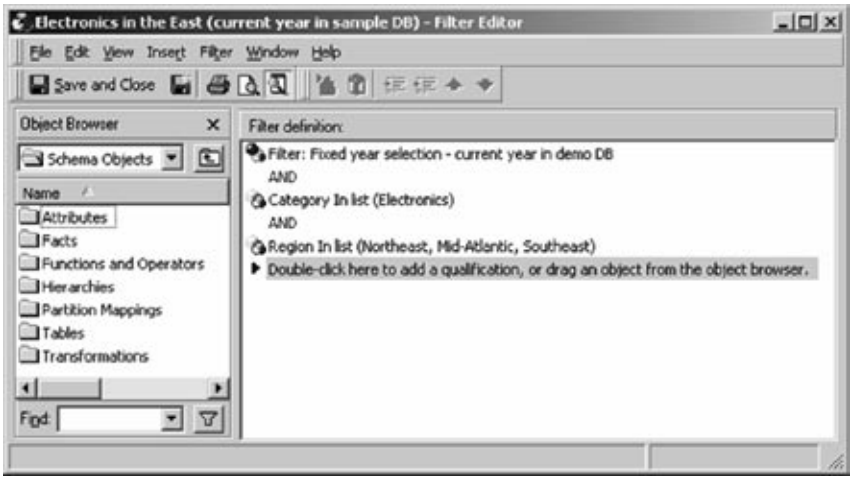


Figure 15.14 Filters control the amount of data that is included in reports.

MicroStrategy Desktop is the only product in the market that allows users to create advanced analyses in a fully integrated, object-oriented environment. Users can identify commonalities from the most detailed, transaction-level data to perform advanced set-based reporting. These sets are completely integrated into the metadata, allowing users to re-use existing analysis objects.

Report developers can define sophisticated business logic for customer segmentation. For example, in order to identify customers who have a high propensity to purchase new televisions, a marketing analyst can identify a set of customers who have purchased TVs. Using MicroStrategy, a separate set identifies products most often purchased by customers who buy the TVs. A third set identifies customers who have purchased the related products but not the TVs, and use them in the marketing campaign. All this is achieved through a unified metadata and common interface.

- Prompts

  A prompt is used to dynamically modify the contents of a report. With prompts, users can determine, during report execution, the objects they want to retrieve for the report and the report filtering conditions. In addition, users can make different prompt selections each time they run the report.

  The MicroStrategy architecture allows for many types of prompts, including:

  i. Filter definition prompts – prompt types that allow users to define the filtering criteria, such as attributes in a hierarchy, attribute forms, lists of attribute elements, and metrics

ii. Object prompts – displays MicroStrategy objects for selection in a report, such as attributes, metrics, and custom groups

iii. Value prompts – captures a single input value, such as a date, a number, or a text string. The value given is compared to metrics or attributes to filter the data.

iv. Level prompts – allow users to specify the level of calculation for a metric

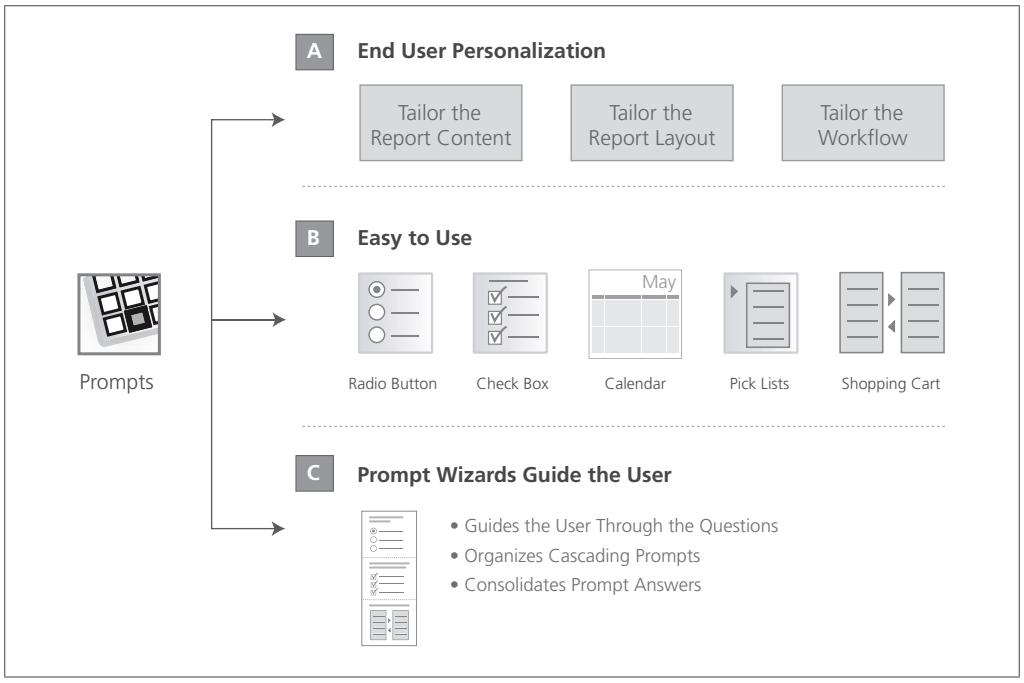
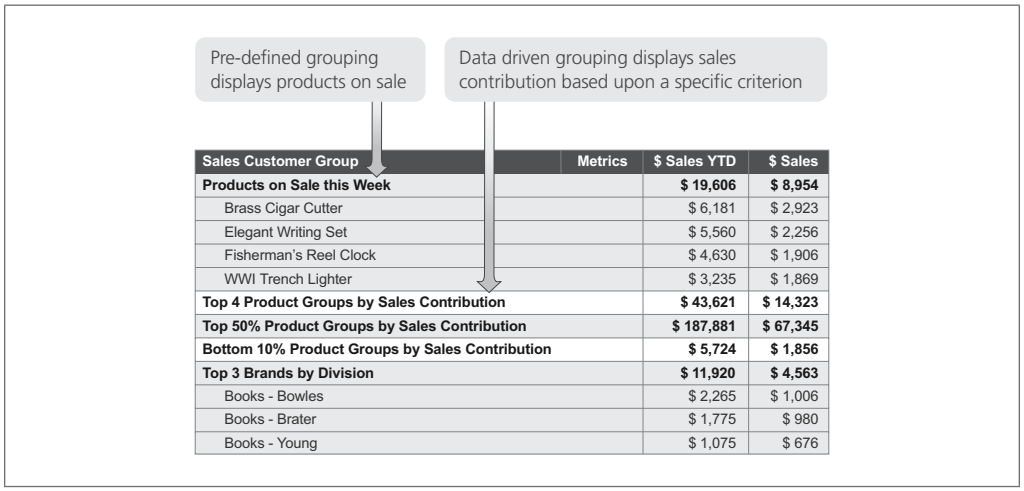


Figure 15.15 Prompts personalize the user experience at report runtime.

- Custom Groups

  A custom group is an object that can be placed on a report layout template, and is made up of a collection of elements called custom group elements. Each element contains its own set of filtering or banding qualifications.



Sales Customer Group	Metrics	$ Sales YTD	$ Sales
Products on Sale this Week		**$ 19,606**	**$ 8,954**
Brass Cigar Cutter		$ 6,181	$ 2,923
Elegant Writing Set		$ 5,560	$ 2,256
Fisherman's Reel Clock		$ 4,630	$ 1,906
WWI Trench Lighter		$ 3,235	$ 1,869
Top 4 Product Groups by Sales Contribution		**$ 43,621**	**$ 14,323**
Top 50% Product Groups by Sales Contribution		**$ 187,881**	**$ 67,345**
Bottom 10% Product Groups by Sales Contribution		**$ 5,724**	**$ 1,856**
Top 3 Brands by Division		**$ 11,920**	**$ 4,563**
Books - Bowles		$ 2,265	$ 1,006
Books - Brater		$ 1,775	$ 980
Books - Young		$ 1,075	$ 676

Figure 15.16 Custom Groups define the content of rows in a MicroStrategy report.

For example, using the Custom Group Editor, a custom group for Store Inventory can be created as follows:

- Small stores with low inventory: Store Sales < 50 AND Store Inventory < 200
- Large stores with low inventory: Store Sales > 50 AND Store Inventory < 200

Custom groups provide drill down capabilities to their constituent attribute elements, and can appear on the report in the following format.



Figure 15.17 Custom Groups can display their constituent attributes in MicroStrategy reports.

- Consolidations

  Consolidations group specific attributes together. They allow developers to place the grouping of attribute elements on a template just like an attribute. The elements of the consolidation appear as rows on the report, and they can have arithmetic calculations.

  For example, suppose users wanted to see each season as a separate row on a report, but Season does not exist as an attribute in the project. A consolidation allows you to group together the elements of the Month of Year attribute into various seasons, and place them on the template. This consolidation will contain four consolidation elements, one for each season. Summer consists of June + July + August; Fall consists of September+ October + November, and so on. The consolidation is placed in the rows of the report with the desired metrics in the columns. Therefore, when the user runs the report, the metric values for June, July, and August are added together to yield the value for Summer. This occurs for each season.

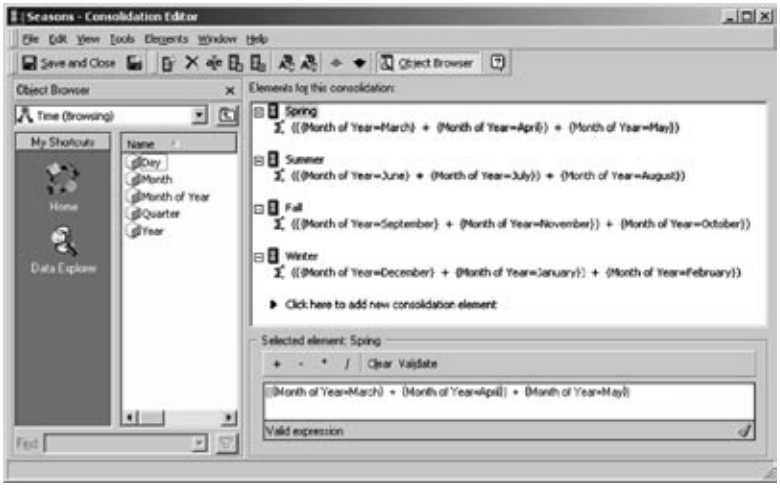


Figure 15.18 Consolidations enable arithmetic between individual rows or groups of rows in a report.

15.6 Report Objects

The Application objects described above are usually developed by a core group of developers who create re-usable objects with standard definitions that can be made available to all applications and users. For example, organizations need to maintain a standard definition of the metric "revenue," which should be common across all users and reports. Report development in the MicroStrategy platform is usually carried out by a larger population comprised of IT developers and report users.

There are two types of Report Objects in the MicroStrategy BI platform:

1. Grid and Graph Reports – These are single data set reports that are displayed to users either in a cross tab grid format or defined graph formats.
2. Report Services Documents – These are used to create scorecards, dashboards, invoices, statements, and other operational and business reports which require custom formatting and print perfect layouts. These may be comprised of one or more data sets from different sources.

A report is a MicroStrategy object that represents a request for a specific set of formatted data from the data warehouse, operational database or cube database. Reports are the focus and goal of business intelligence. They allow users to gather business insight through data analysis.

Reports in MicroStrategy can come from the following data sources.

- Data warehouses using MicroStrategy's dynamic SQL engine
- Operational databases and datamarts using freeform SQL
- Cube databases, such as SAP BW, using MicroStrategy's dynamic MDX engine

15.7 Report Creation Using the Dynamic SQL Engine

Most MicroStrategy reports are built using fundamental building blocks such as data abstraction objects and business abstraction objects. When these are assembled onto a report, the MicroStrategy's dynamic SQL engine generates optimized SQL at report runtime.

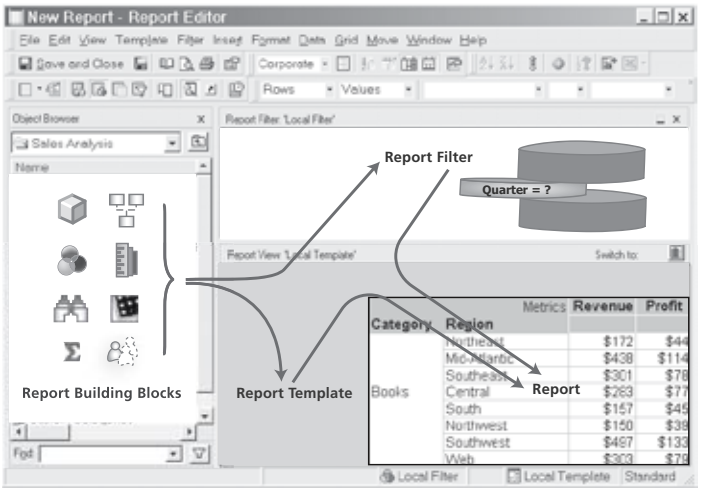


Figure 15.19 Report design in Desktop involves defining the filter (data) and template (layout) from the report abstraction objects in the metadata.

Reusable templates specify the layout and formatting of the data on a report. Attributes, hierarchies, metrics, custom groups, consolidations, and prompts can be placed in rows or columns on the template grid, or in pages to break reports into manageable pieces. Report developers can specify data presentation characteristics such as font, color, alignment, and number formats, as well as additional report characteristics such as subtotaling, metric thresholds, and graph properties. Filters specify the conditions that the data must meet to be included in a report or in a metric.

Reports and documents present decision makers with information gleaned from the organization's data stores. Report developers can assemble all analysis objects necessary to turn data into insight using the most intuitive presentation format – whether it be a classical cross-tabulated grid, a graph, or a highly visual dashboard or scorecard.

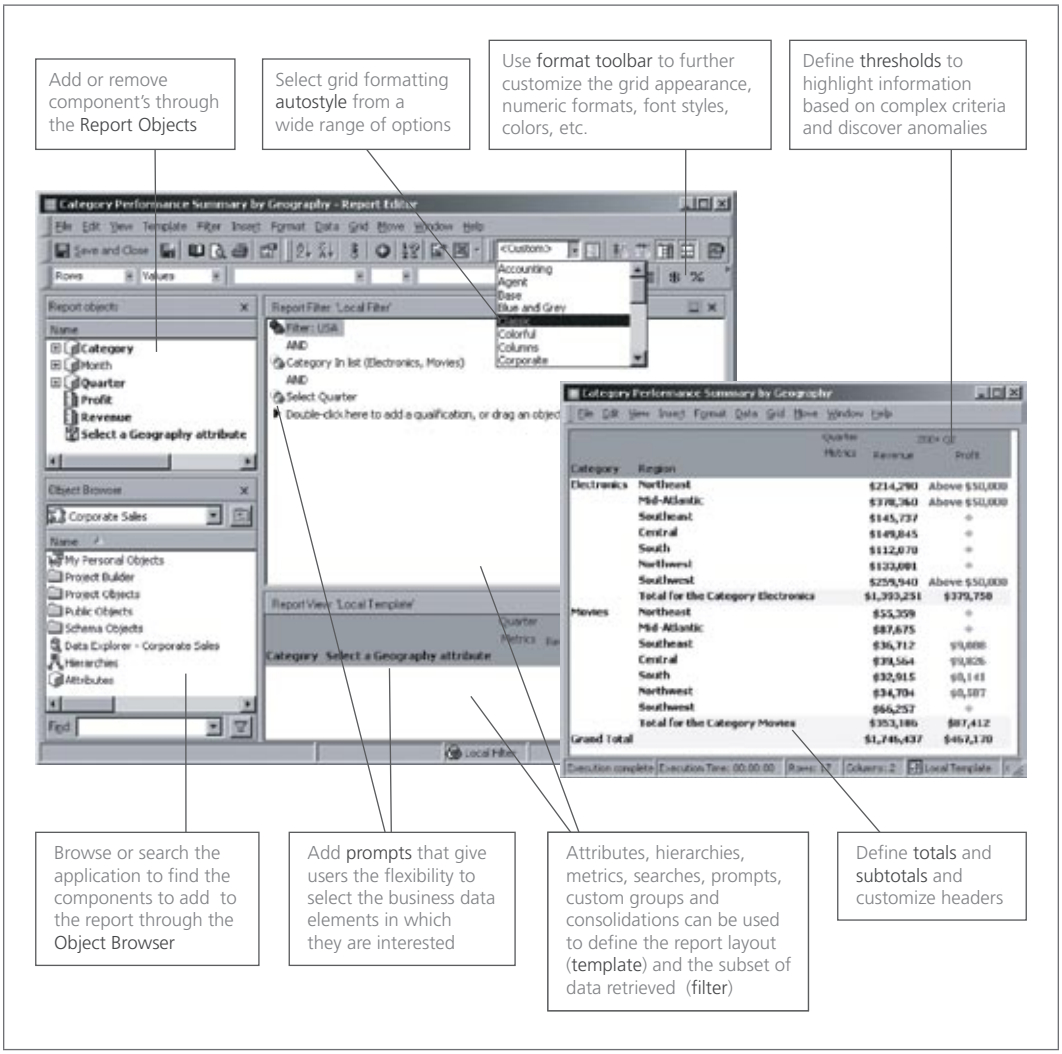


Figure 15.20 Report developers have a wide range of features to create any report or analysis.

Tuning Report SQL using VLDB Properties

VLDB properties allow developers to customize the SQL that MicroStrategy generates. Developers can configure settings such as SQL join types, SQL inserts, table creation properties, and so on. Developers can use the VLDB Properties Editor to alter the syntax of a SQL statement, and take advantage of unique, database-specific optimizations.

Some of the qualities that make these properties valuable are:

- Optimization: Take advantage of database-specific settings to further enhance the performance of queries.
- Flexibility: VLDB properties are available at multiple levels so that the SQL generated for one report, for example, can be manipulated separately from the SQL generated from another, similar report.
- Complete database support: VLDB properties allow you to easily incorporate, and take advantage of new database platforms and versions.

VLDB properties can be set at various levels in the MicroStrategy architecture. The following diagram shows how VLDB properties that are set for one object take precedence over those set for another.

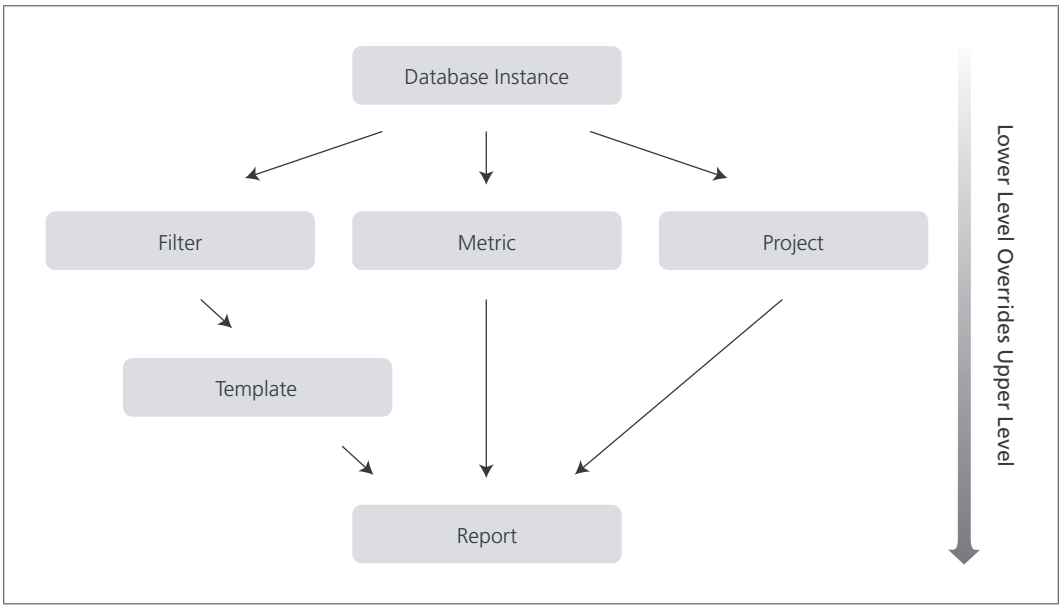


Figure 15.21 VLDB Order Priority.

15.8 Creating OLAP Services Cube Reports

MicroStrategy OLAP Services allows for the creation of Intelligent Cubes. These cubes are cached multi-dimensional collections of attributes and metrics which constitute a sandbox of information within which users can perform OLAP operations to get speed of thought analysis. However, Intelligent Cubes are not bound by the limitations of traditional cubes. There is no cube loading and maintenance process, and users can drill outside of the cube at any point to retrieve more information from underlying data sources.

Reports using Intelligent Cubes are created in MicroStrategy Desktop in a very similar manner to regular reports. Developers add metrics, attributes, hierarchies, custom groups, and other objects to the layout template view and add any filter criteria.

In addition, for OLAP Services-enabled reports, developers have the option of adding objects to the "Report Objects" pane. Attributes and metrics placed in this window become part of the cached OLAP Services cube. The user can drag and drop or drill to these available objects at any time, and retrieve the information from the OLAP Services cube.

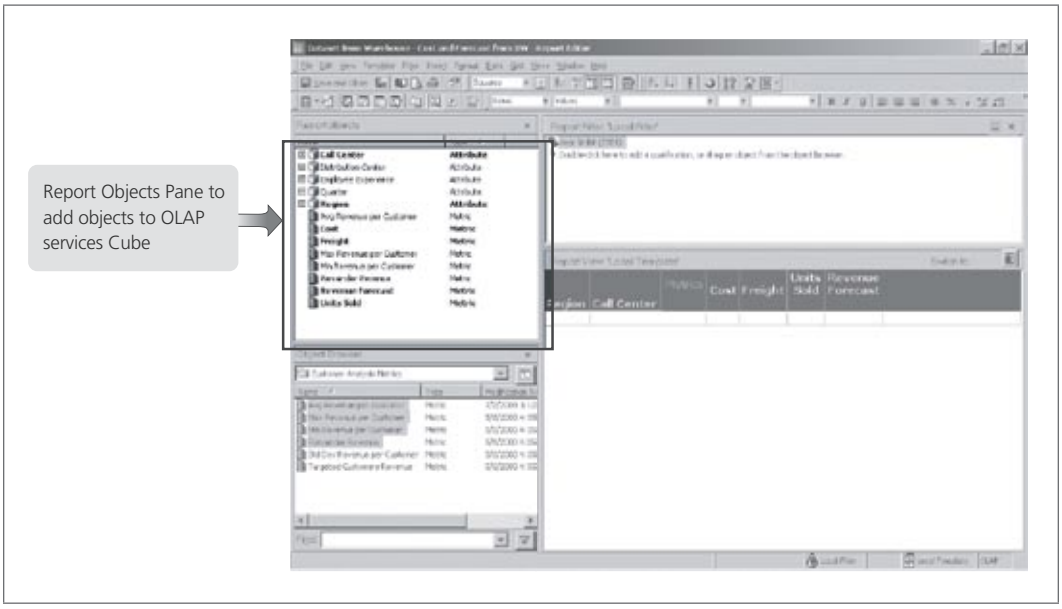


Figure 15.22 Report Designer for OLAP services-enabled reports

15.9 Build and Execute Reports with the Dynamic MDX Engine

The MicroStrategy BI platform extends the scope of analyses provided by BI applications by incorporating data assets other than the traditional data warehouse. Organizations often have information stored in multiple locations, such as SAP BW and Microsoft Analysis Services cubes. These cube databases can be queried using the MDX query language.

Intelligence Server has a dynamic MDX engine that allows organizations to query the information stored in these databases, and also combine this information with other data sources using Report Services. The Dynamic MDX engine creates MDX based on user query requests at runtime, similar to the SQL engine.

For example, the multi-dimensional models of SAP BW InfoCubes and QueryCubes are imported and represented in the data abstraction objects. This import process allows MicroStrategy to read all the SAP BW business content such as queries, InfoCubes, variables, structures, hierarchies, key figures, and characteristics. These SAP metadata elements are translated into corresponding MicroStrategy metadata objects. SAP BW cube objects are mapped to business abstraction objects, and used to create reports and documents.

During Report execution, the MicroStrategy metadata serves as a pointer to the underlying SAP metadata. MicroStrategy dynamically generates optimized MDX, and executes the MDX against SAP BW to retrieve the data.

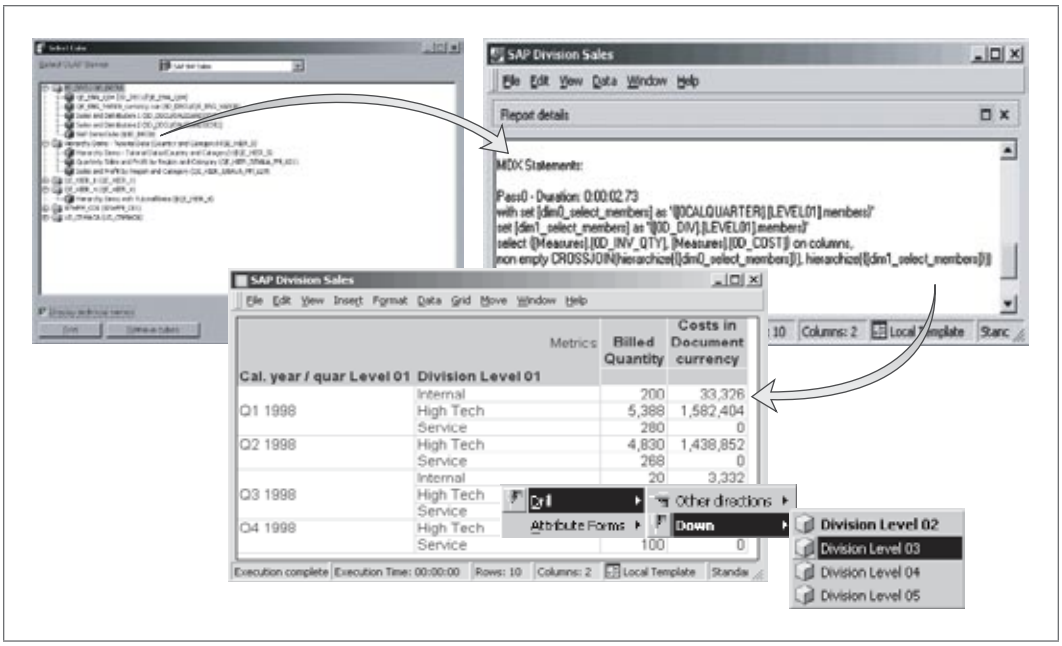


Figure 15.23 Reports can be built against cube databases such as SAP BW cubes.

15.10 Operational Reports Using the Freeform SQL Engine

The Freeform SQL functionality adds great flexibility to MicroStrategy's query and reporting capabilities. Traditionally, developers use the MicroStrategy Engine to generate SQL to run against one specific relational database to get a desired report. In addition to generating reports in the traditional way, developers can also use their own customized SQL statements to generate reports from operational systems available in the organization. This capability saves organizations a tremendous amount of time since developers do not need to place the data into a data mart or data warehouse first. In addition, these reports may be used in cases where quick proof of concept reports need to be built or in cases where the table structures in the data sources are not ideally suited for engine SQL generation.

The MicroStrategy Freeform SQL editor is flexible to allow developers to insert prompts and security filters into the SQL statement so that the results that are returned are personalized for each user.

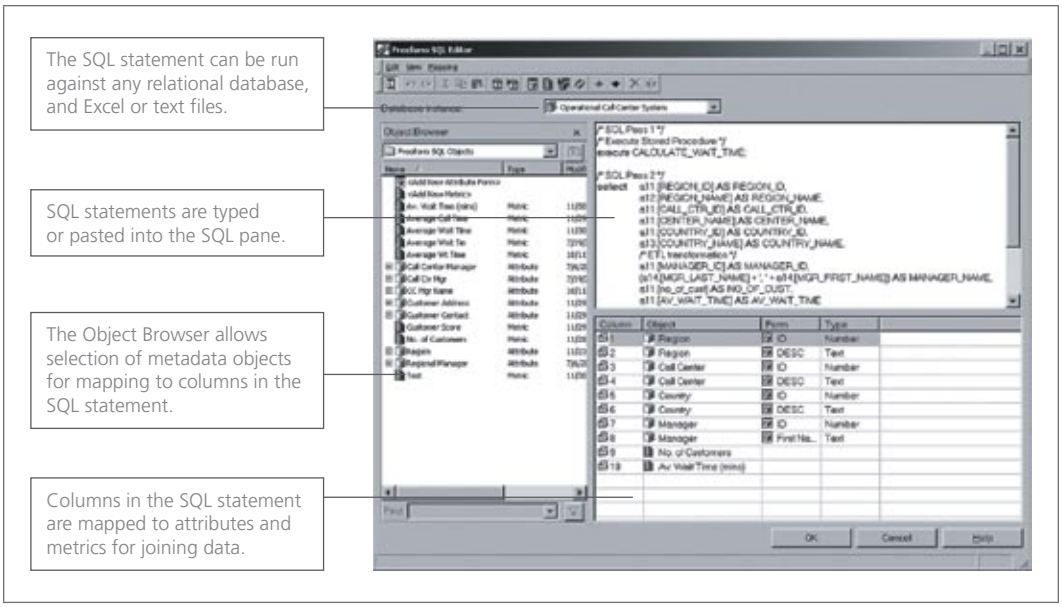


Figure 15.24 Freeform SQL editor in MicroStrategy Desktop.

15.11 Building Graphs in MicroStrategy Desktop

Once a report has been laid out using a template and filter combination, the report developer can also choose to expose the report as a graph. MicroStrategy Desktop provides a comprehensive set of graph formatting options to allow users to graphically visualize reports in the best possible manner.

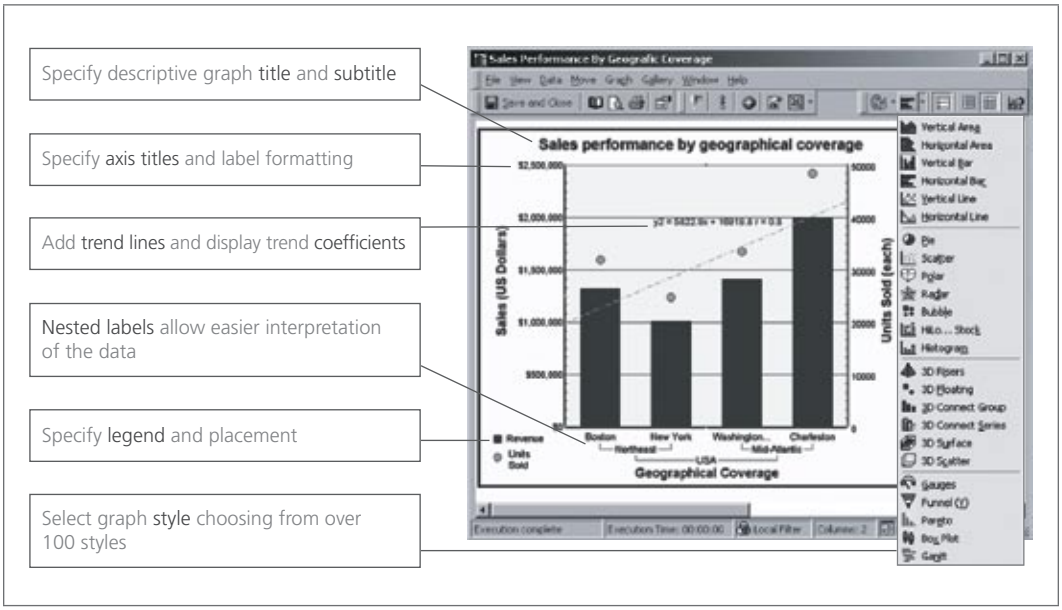


Figure 15.25 Graphs provide the means to show visually appealing and effective graphical representations of the data.

15.12 Data Mining and Predictive Reports in MicroStrategy Desktop

Using MicroStrategy Desktop, report developers can create multi-variant linear regressions, train the regression model based upon live data, and use it to create forecasts of future events.

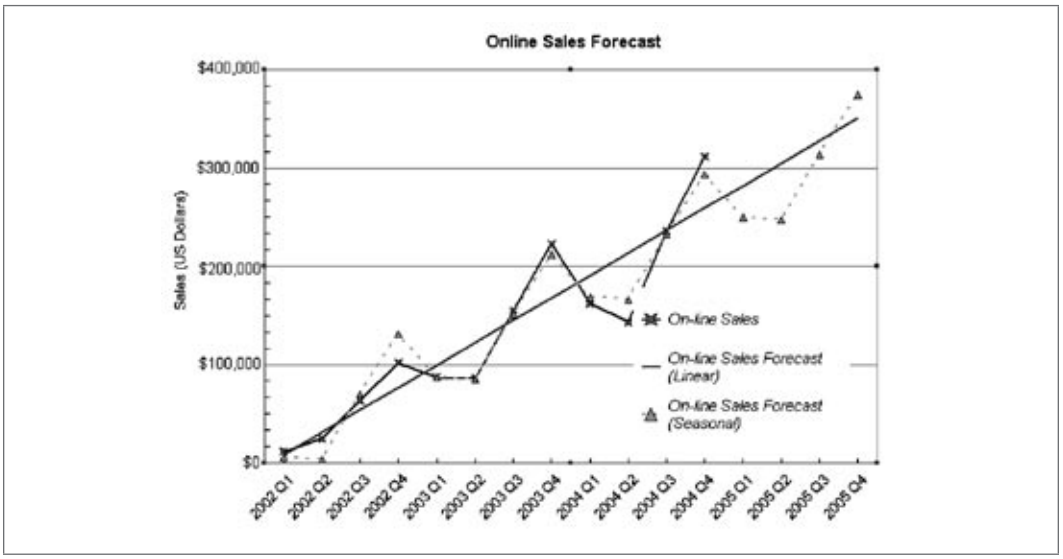


Figure 15.26 The above example shows the forecast based upon seasonal variation in online sales (dotted line), which provides a more realistic prediction as compared with the straight-line linear regression option (straight line).

Organizations can leverage their investments in third-party data mining tools by importing predictive models created in those tools in Predictive Modeling Markup Language (PMML), and use the predictive metrics in any report.

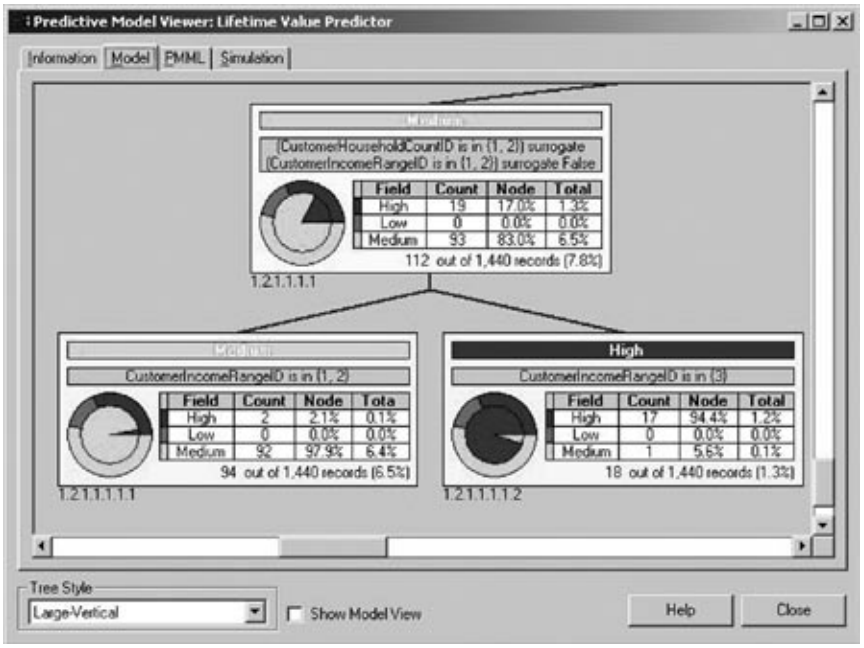


Figure 15.27 This example shows the model view of a decision tree algorithm that was imported into MicroStrategy. Such a model, for example, can be used effectively for customer segmentation analysis.

15.13 Report Services Document Creation

MicroStrategy Report Services raises the bar for enterprise reporting by combining the best OLAP analysis and predictive analysis capabilities with pixel-perfect enterprise reports, scorecards, and dashboards on a unique industrial-strength business intelligence platform.

Report Services Documents in MicroStrategy are used to create print perfect and pixel perfect document layouts. Types of documents that can be created with MicroStrategy Report Services are:

- Scorecards
- Dashboards
- Business performance management reports
- Invoices and Statements
- Operational reports
- Compliance and Auditing reports

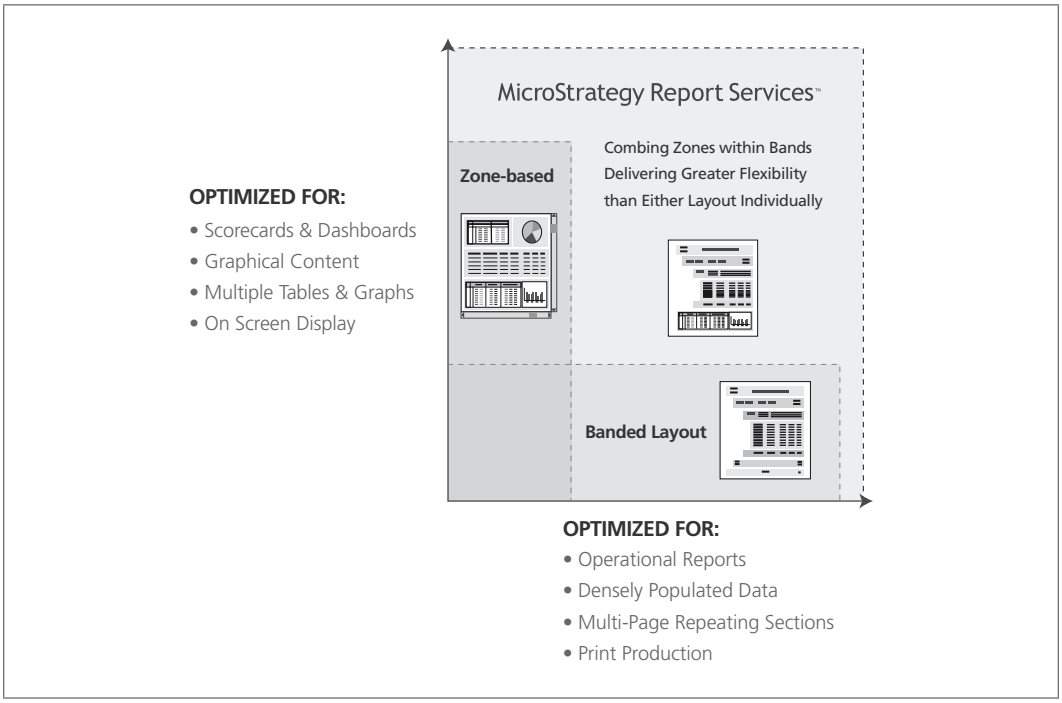


Figure 15.28 MicroStrategy combines banded and zone-based layout to deliver the most flexible reporting environment.

Report Services Layouts

Banded Layout is the most familiar report layout technique today because it is the predominant vehicle for operational reporting. It has been the cornerstone of all leading report writers for years. The reason for its long-term popularity for Operational Reporting is that it allows the organization of large amounts of data into a natural hierarchy of information that users can easily navigate. The term "banded" refers to specific horizontal bands superimposed on the report layout screen that dictate where data should be automatically summarized for page headers and footers, report headers and footers, and within user-defined groupings or hierarchies that map to the business organization.

Banded report layout is truly optimized for traditional operational reporting, with its dense hierarchical information presentation, multi-page repeating sections that cover entire ranges of enterprise operations, and need for fine printing optimization. MicroStrategy offers pure banded layout following the same visual paradigms that are now well understood by any person who has created enterprise reports using any of the common report writing products available today.

Zone-based Layout is optimized for the creation of scorecards and dashboards. The term "zone" refers to the graphical building block technique that allows users to layout entire tables and graphs onto a page or screen, where each of these tables or graphs has a presentation behavior of its own. Zone-based layout is optimized for displaying graphical content rather than tabular data. Zone-based layout is further optimized for on-screen display, where users have scroll bars to move around the report, and high quality printing is not a requirement.

MicroStrategy users can build scorecards and dashboards on the fly using simple drag-and-drop techniques to insert multiple reports, graphs, text, hyperlinks, and images onto the report layout screen. These objects may also be arranged anywhere in the layout, and automatically adapt to the size and shape of their content, moving other objects dynamically to fit. The MicroStrategy Development paradigm allows for the support of both zone-based and banded report layout paradigms.

Creating Report Services Documents in MicroStrategy Desktop

Report Services Documents are created in MicroStrategy Desktop using the Document editor. The Document Editor allows developers to create, customize, and save documents to be used across the MicroStrategy platform. There are four main sections in the editor:

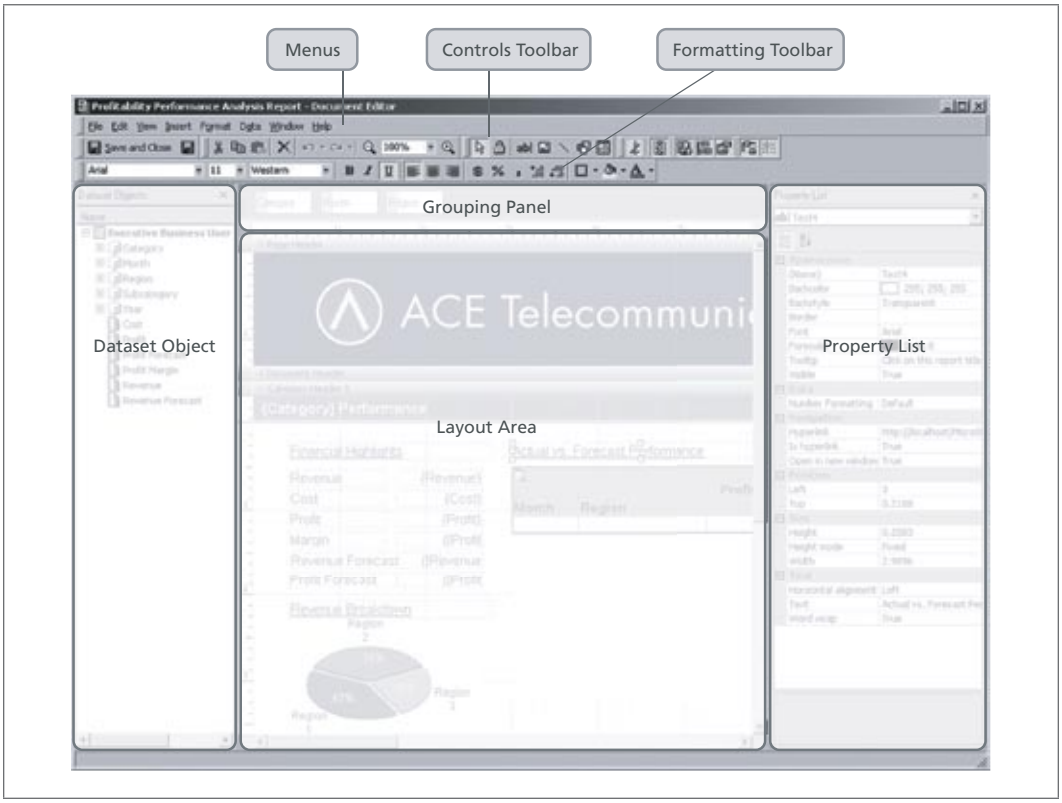


Figure 15.29 MicroStrategy Desktop Document Editor Layout.

1. Dataset Objects

   This section displays all of the datasets used in the document. Datasets are any MicroStrategy Report, and can come from any source such as data warehouses, operational databases or SAP BW. Users can combine data from multiple sources into a single document by adding multiple datasets. It lists all attributes, metrics, custom groups, and consolidations in the existing MicroStrategy report, regardless of whether or not they are displayed on the report.

2. Layout Area

   The Layout area provides the framework for precisely controlling the display section of the fields when the document is viewed as a PDF, Excel or in HTML view. To add data, developers drag objects from Dataset Objects, and drop them into this area. Depending on the section they place controls in, they print or display differently.

   - Page Header/Footer: prints at the top and bottom of each page
   - Document Header/Footer: prints at the beginning/end of the document
   - Grouping Header/Footer: for each field in the Grouping panel, prints before and after the Detail Header Footer
   - Detail Header/Footer: prints immediately before and after each group of Detail sections
   - Detail: repeats for each row in the dataset

3. Grouping Panel

   The Grouping panel lets developers group information in the document in a hierarchical structure. To add a group, developers drag and drop any attribute, consolidation, or custom group from the Dataset Objects onto the Grouping panel. The sequence automatically determines the hierarchy.

4. Property List

   The Property List displays the properties of the object selected on the Layout area. The properties that are listed vary depending on the type of object (text field, image, line, section, and so on) selected.

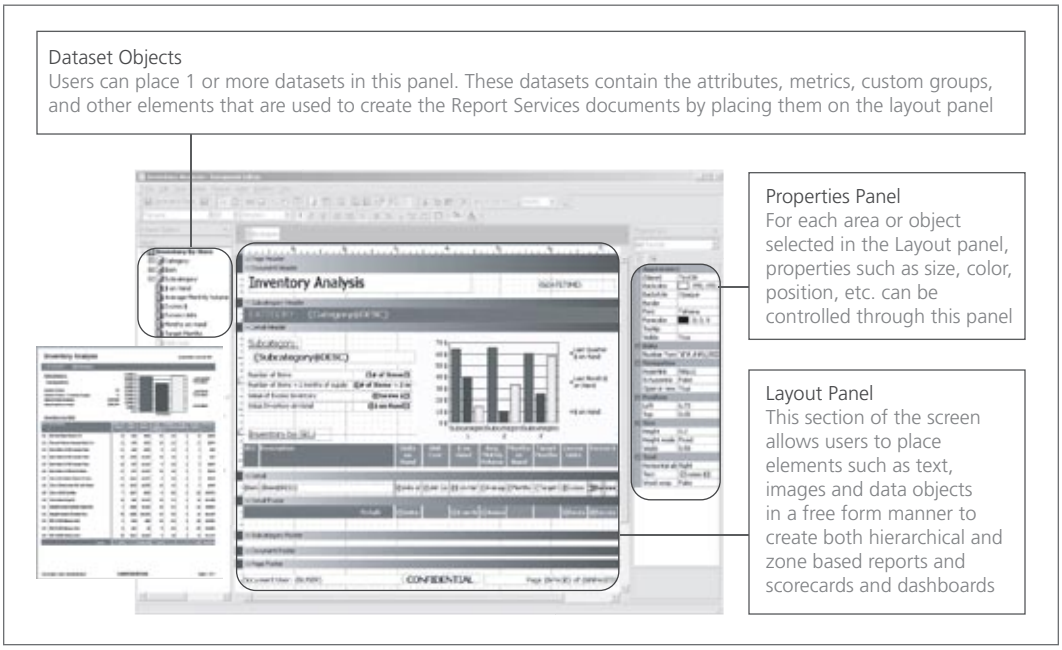


Figure 15.30 Illustrated example of creating a Report Services Document in Desktop.

15.14 Web Professional

As BI applications grow in scope and usage, two important themes regarding the user population have become apparent:

1. Traditional lines between roles such as end user, power user, and developer are rapidly becoming blurred, with end users requiring more functionality and ad hoc capabilities to create their own reports
2. Organizations have distributed development teams spread out across different business units and geographies. Applications usually have core key figures and characteristics defined by an architect, but report development and creation needs to be farmed out to a larger user population

Installing a thick client desktop application on end user machines is too expensive to deploy and maintain for larger user populations. A true thin client Web interface that allows for the editing of reports and the creation of new reports is required. MicroStrategy Web Professional offers organizations and developers just that capability. The main features of MicroStrategy Web Professional from a developer perspective are:

- Create new Grid and Graph reports over the Web using a thin client browser interface
- Create pixel-perfect formatted dashboards, scorecards and operational reports over the Web
- Edit existing reports and dashboards over the Web in a WYSISYG manner

Ability to Create Grids and Graphs Over the Web

MicroStrategy Web Professional offers Web-based developers and power users the ability to create reports in the form of grids and graphs.

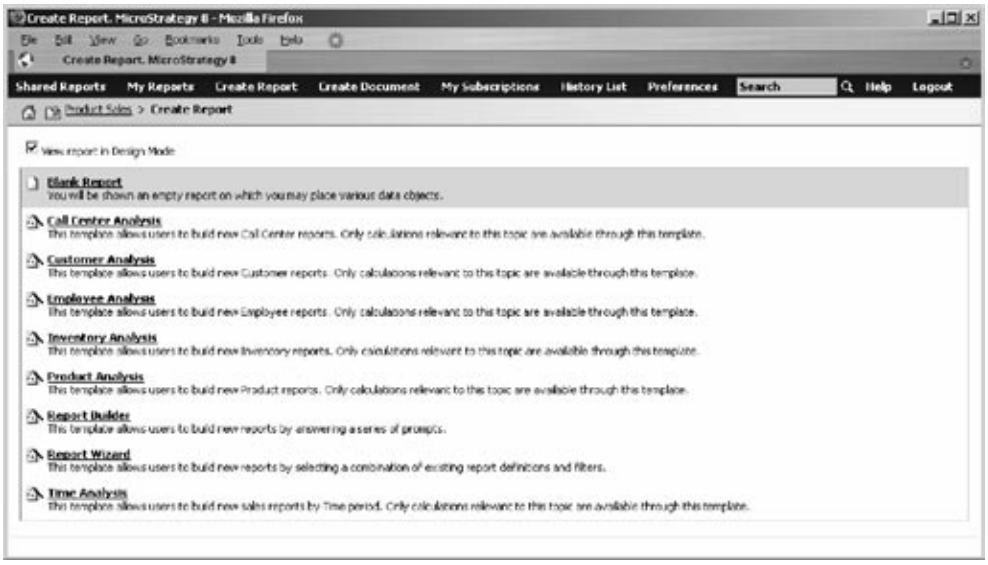


Figure 15.31 The report creation options in the MicroStrategy Web interface.

There are a number of options with which a user can create reports in MicroStrategy Web.

- Ad Hoc Report Creation from a Blank Template
  This option allows users to pick and choose from any available metrics and attributes within the MicroStrategy metadata. Users drag and drop these objects on the design area to create the report. Complete ad hoc report creation is possible with the developers choosing from any available objects in the MicroStrategy metadata.

There are four steps to creating a report in MicroStrategy Web Professional using a blank template.

i. Object Browser Panel – Choose from all the attributes and metrics available in the MicroStrategy metadata to place on the report template

ii. Report Filter Window – Developers and power users add conditions to restrict the data brought back to the user. Existing filters can be reused or new filters can be defined from the objects in the Browser Panel.

iii. Page-by Axis for drop down attributes – Users can also place attributes and custom groups into page by axis. When attributes are placed in this position, they generate a drop down list of elements that users can scroll through. These can be moved back to the main report window once the report is run if needed.

iv. Template definition pane built using drag and drop – Users drag and drop attributes and metrics from the Object browser pane on to the template area. This allows users to choose order, placement, and location of attributes and metrics on the report layout.

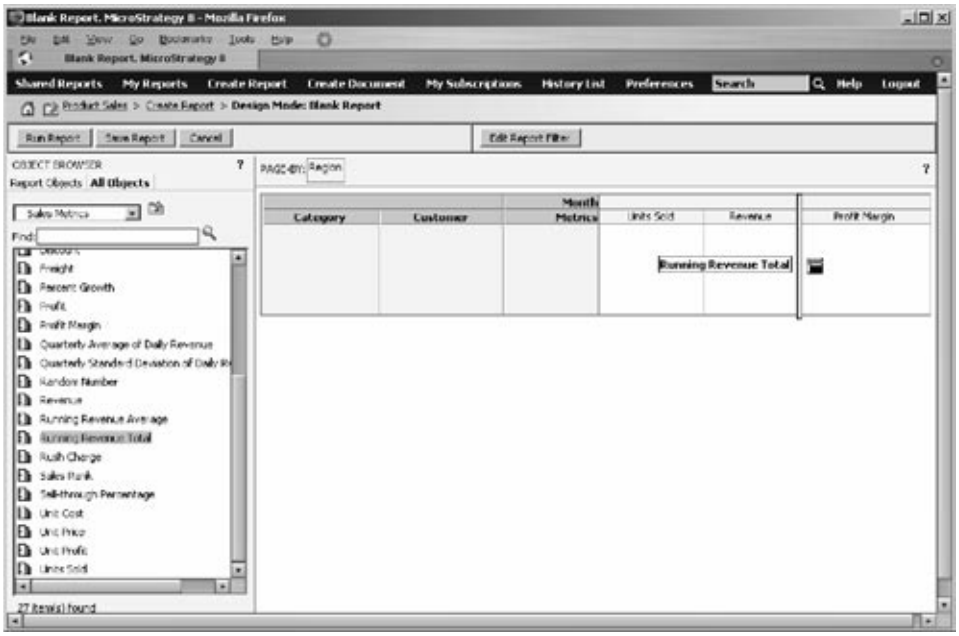


Figure 15.32 Ad hoc report creation in MicroStrategy Web using a blank template.

- Ad Hoc Report Creation from a Predefined Template

  A predefined template contains the set of associated metrics and attributes stored in an Intelligent Cube. This option narrows the range of objects presented to the user. For example, users can then choose from a subset of objects in a closely related area such as customer analysis or financial analysis.

  Once users choose a particular template, they can drag and drop the attributes and metrics contained in that template, and create reports in a manner similar to the ad hoc report creation process.

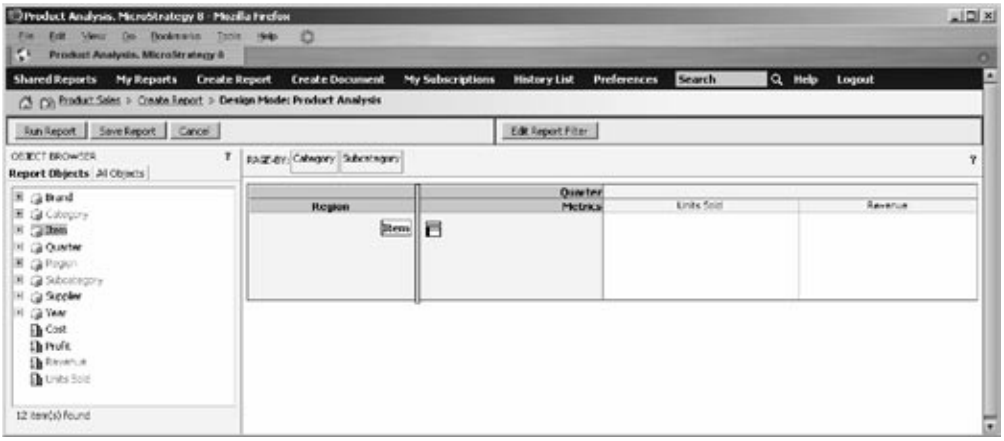


Figure 15.33 Ad hoc report creation in MicroStrategy Web from a predefined template.

- Using the Report Builder
  This option presents a set of prompts that enable the selection of attributes, metrics, hierarchies, and conditions needed for a particular report in an ad hoc fashion. Report Builder is useful for users who require a tightly controlled workflow for report creation.



Figure 15.34 The Report Builder prompts for report objects and filtering criteria.

A series of four prompts asks for:

i. Attributes to view on the report
ii. Metrics to view on the report
iii. Filtering criteria on any attribute
iv. Filtering criteria on any metric

- Using the Report Wizard
  Users can create reports by choosing a combination of existing filters and templates. Templates define the layout of the report while Filters define the conditional criteria used to narrow the data. Users have the flexibility to choose any available filter-template combination to create the report.

This option may be used in cases where developers need limited flexibility and tight control in report creation.

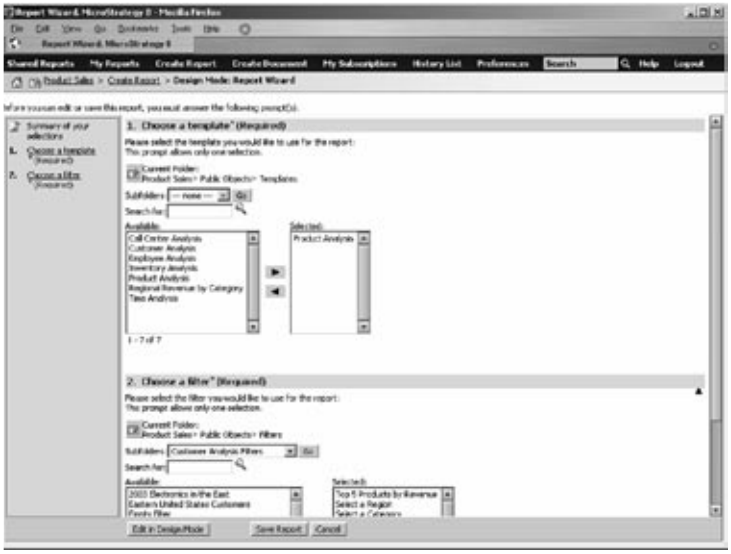
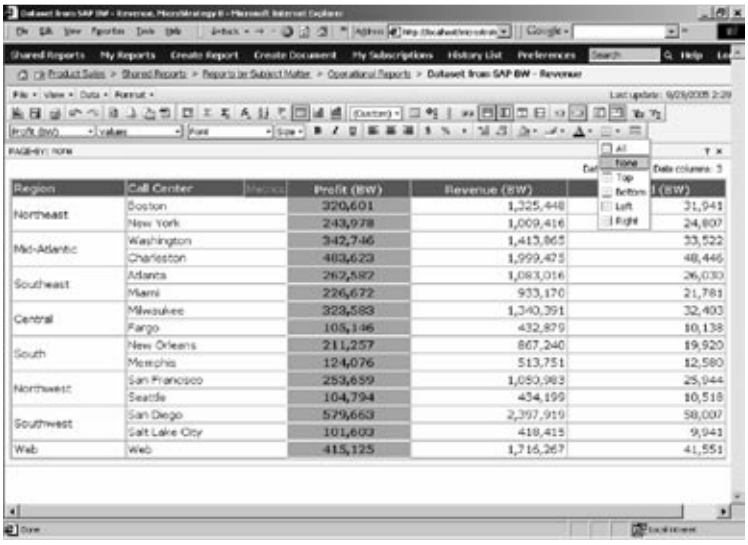


Figure 15.35 The Report Wizard prompts for templates and filters stored in the metadata.

Fine-Grained Control of Grid Formatting in MicroStrategy Web

Once the report has been created, MicroStrategy Web offers powerful abilities to the user to format the report in either grid or graph formats for print or presentation purposes. With sophisticated formatting toolbars and functionality intuitively familiar to users of spreadsheet applications, the properties of various cells in a report can be changed to achieve the desired grid formatting. Colors, fonts, styles, numbers, alignment, borders, and sizes can be altered by the users over the Web.

These properties can be accessed either from a readily available formatting toolbar or by right clicking on any attribute or metric to get to the formatting options for that particular set of values.



Region	Call Center	Profit (BW)	Revenue (BW)	(BW)
Northeast	Boston	320,601	1,325,448	31,941
	New York	243,978	1,009,416	24,807
Mid-Atlantic	Washington	342,746	1,415,865	33,522
	Charleston	483,623	1,999,475	48,446
Southeast	Atlanta	262,582	1,083,016	26,030
	Miami	226,672	933,170	21,781
Central	Milwaukee	323,583	1,340,391	32,403
	Fargo	105,146	432,879	10,138
South	New Orleans	211,257	867,240	19,920
	Memphis	124,076	513,751	12,580
Northwest	San Francisco	253,659	1,050,983	25,944
	Seattle	104,794	434,199	10,518
Southwest	San Diego	529,663	2,397,919	58,007
	Salt Lake City	101,603	418,415	9,943
Web	Web	415,125	1,716,267	41,551

Figure 15.36 Grid formatting with MicroStrategy Web.

Display Report Information in Graphical Format in MicroStrategy Web

Users often need to visualize data in graphical formats. A well designed graph can quickly convey more information than a large grid of information by using visual aids to contrast areas of interest or concern . MicroStrategy Web allows users to quickly display grid reports in graph format using a simple one click menu action.

MicroStrategy Web offers developers and power users the ability to quickly change the formats of graphs that they create, or to edit existing reports.

Business users have the power to format the graphs exactly as they want, without having to rely on IT developers. Through the graph formatting toolbar and dialog boxes, many graph properties can be adjusted, including:

- Graph types
- Graph Subtypes
- Legends
- Axes Scales and Labels
- Fonts
- Series and Category options

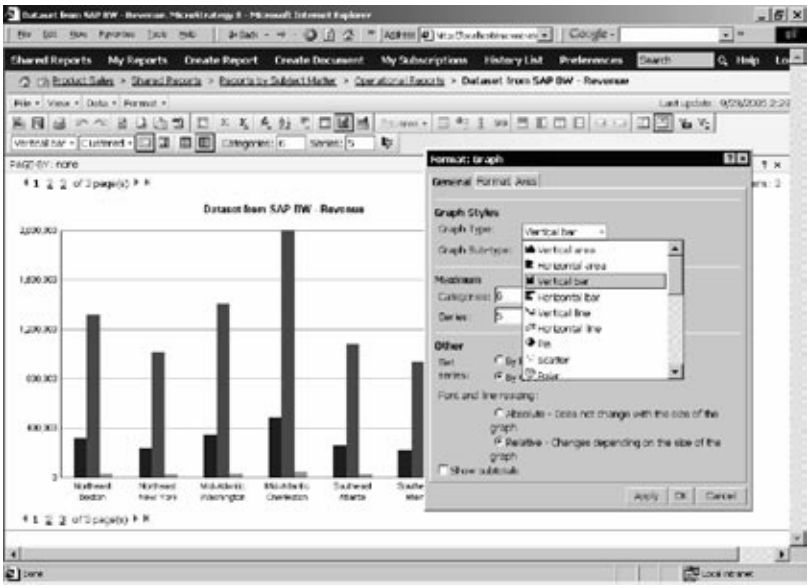


Figure 15.37 Powerful graph formatting over the Web in MicroStrategy Web.

15.15 Creating Pixel-Perfect Formatted Dashboards and Scorecards Over the Web

In addition to creating grid and graph OLAP reports, well formatted Report Services documents can be defined in MicroStrategy Web. Developers can add datasets, drag and drop fields, insert grids and graphs, and use powerful formatting to create print-perfect scorecards, dashboards and operational reports. The editors and techniques used to create these reports are similar to the Desktop Report Services Document editor.

Users have the option of choosing either a Blank template or any other existing corporate template as a base for the creation of these documents. By providing a set of standard corporate templates, developers control common fonts, headers, footers, and other corporate formatting standards.

Datasets are then added to the document. Any MicroStrategy report, regardless of where the data is retrieved, can be used in a Report Services document. If more than one dataset is used, the information is automatically merged, and metric calculations across datasets are possible.

All attributes and metrics in the datasets are added to the document by dragging and dropping them on to the layout template. They can be placed in stand alone fields in the detail section, and also into various bands and zones as defined by the document grouping. Precise control of the object placement and size defines the appropriate formatting and layout required. The documents are saved into the standard folder structure for other users to access.

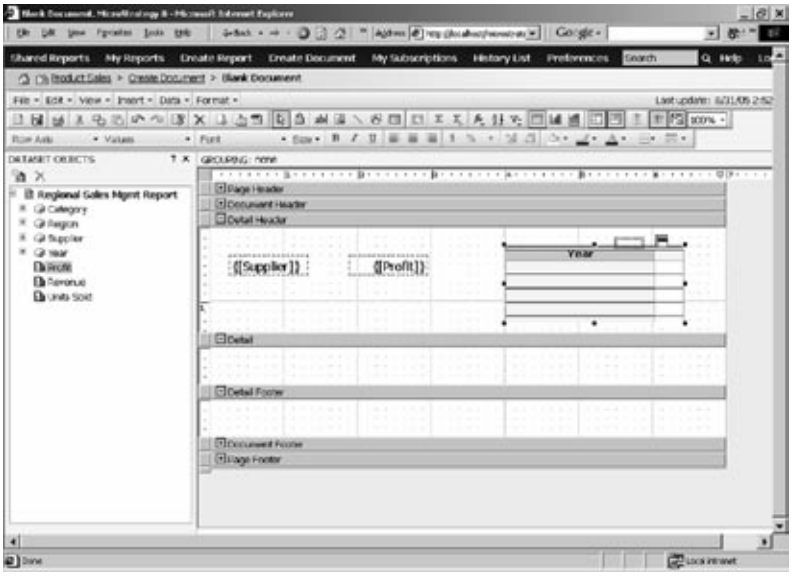


Figure 15.38 Formatting and layout of Report Services documents in the document editor in MicroStrategy Web.

15.16 WYSIWYG Editing of Existing Reports and Documents Over the Web

In addition to the creation of new documents and scorecards, developers and users have the ability to edit these documents in a WYSIWYG manner over the Web. Users can change formatting, layout, and other parameters of a report to tailor the layout and contents to their needs. This drastically reduces report development times by eliminating endless iterations with report developers in IT departments.

MicroStrategy Web dramatically increases the number of users who have access to powerful ad hoc as well as WYSIWYG report editing and creation abilities. The interface is intuitive, and is used with minimal training by any user. In conjunction with a secure architecture and a true thin client interface, this presents an opportunity to distribute development teams geographically and by business units, reducing the number of IT developers needed to maintain the application.

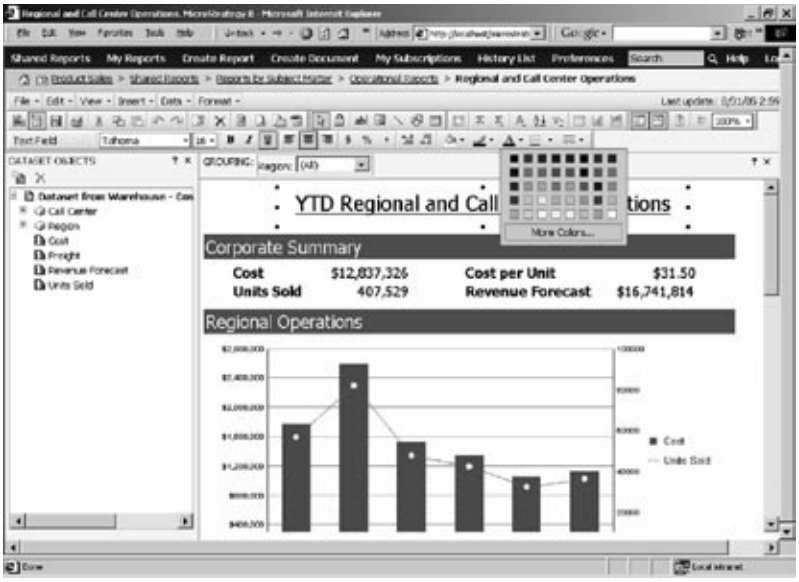


Figure 15.39 WYSIWYG Editing changes the layout and format of Report Services Documents directly.

15.17 Narrowcast Server

MicroStrategy Narrowcast Server enables organizations to build sophisticated alerting and batch reporting applications. This section will focus on the developer interfaces available to build these applications.

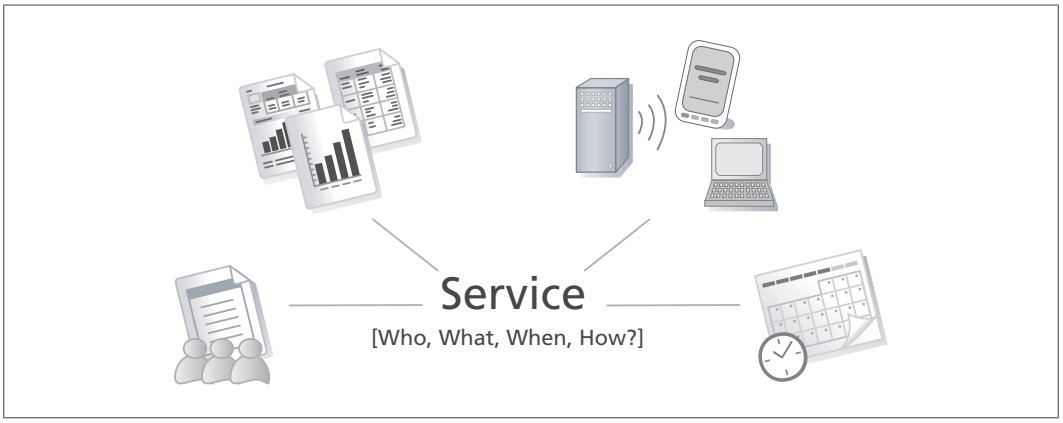


Figure 15.40 Narrowcast Services define the "Who, What, When, and How" for report deliveries.

The end user interfaces for Narrowcast Server express themselves primarily through the MicroStrategy Web Interface and a subscription portal for Web-based message deliveries. In addition, end users receive messages via email, file servers or print servers. The Narrowcast Design Interface is used to develop and deploy alerting and batch reporting applications.

Narrowcast Server Designer

The Narrowcast Server Designer is the primary developer interface for MicroStrategy Narrowcast Server. Through the interface, developers can create the following objects:

- Information Objects
- Users
- Deliver Methods
- Personalization Criteria
- Services
- Schedules (Alert and Exception Driven)
- Subscription Sets (Static or Dynamic)
- Publications
- Documents
- Question Objects



Figure 15.41 New report delivery services are developed in the Narrowcast Server Designer interface.

15.18 Narrowcast Server Design Process

The Narrowcast Server Design Interface has a familiar Windows Explorer navigation paradigm with folders, subfolders, and objects within them. The steps that Narrowcast Server developers go through to create push-type applications can be understood using the following diagram:

Who?	Who will receive this information?	Subscription Set
What?	What information will be sent?	Publication, Document, Information Object, Question Object
When?	When with this information be sent?	Schedule
Where?	Where will the user receive the information?	Device
How?	How will this information be formatted?	XSL File, Locale

Figure 15.42 Steps needed to define a Narrowcast service.

The following section explains each of these in more detail.

1. Services

   A service is the fundamental tool that delivers information. A service is the entity that defines who should receive content, what content they should receive, how they should receive it, and when they should receive it. When developers define a service, they answer these questions. The Service Wizard is the recommended way to create services. It satisfies needs for most application design requirements. The Service Wizard guides you through the process of designing a service by having you define:

   - By which delivery methods should a service be delivered (e.g., email, wireless, or Web subscription portal)
   - The content that should be delivered, such as an HTML email with several reports or a wireless plain text service
   - The recipients who should receive this information and their preferences
   - The schedule on which the information should be delivered

2. Delivery Methods

   Services can deliver content via email, wireless devices, the Subscription Portal, print devices, and file servers. Each means of delivery is called a delivery method or information transmitter. In Narrowcast Server, each service can be defined to support one or more delivery methods. For example, developers might design a service to be sent to all subscribers via email space or to deliver content to subscribers via the Subscription Portal, and also print servers.

3. Publications

   Each delivery method is supported by a single publication, which specifies the content to be delivered. Since each delivery method corresponds to an information transmitter, such as the SMTP information transmitter for email, each publication is created for a single information transmitter. The information transmitter specifies which content sections the publication provides to the application designer and the type of content each section can accept. For example, the email (SMTP) information transmitter informs publications for the email delivery method that a message subject, message body, and a section for attachments should be provided. The email information transmitter also dictates that the subject can only accept one text document, while the attachments section can accept any number of documents of any type. Other information transmitters specify their own content sections and their own rules for the type of content these sections can accept.

4. Information Objects

   An information object is a set of instructions specifying how to get data from an external information source, such as a relational database, the Internet, or a flat file. Information Objects support the key Narrowcast Server capability of gathering and using information from external sources for message content or subscription set information.

5. Service Content

   Each Service also includes content, which must be defined according to delivery method.

   i. Email (SMTP) delivery method, which contains:
      - subject
      - message body (HTML)
      - attachments: includes Excel, HTML, and text, as well as zip and other imported files
      - execution control (advanced): overrides the default information transmitter settings
      - message body (text)

   ii. Wireless (SMTP) delivery method, which contains:
      - subject

- message body (text)
- attachments: includes Excel, HTML, and text, as well as zip and other imported files
- execution control (advanced): overrides the default information transmitter settings
- message body (HTML)

iii. Print delivery method, which contains:
- cover page: allows one or more text or HTML documents; these are printed in the order they appear in the interface
- documents: requires at least one text, PDF, imported, or HTML document; these are printed after the cover page in the order they appear in the interface

iv. File delivery method, which contains:
- saved documents: requires at least one document of any type; more are allowed
- execution control (advanced): allows only one text document; this document is optional

v. SMS (SMPP) delivery method, which contains:
- message body: required and limited to one plain-text document
- execution control document (advanced): overrides the default information transmitter settings; this document is optional and limited to one plain-text document

vi. Web (Portal) delivery method, which contains:
- document body: can be one HTML or one plaintext document
- attachments: can be any number and any type of documents
- link rendering style: an XSL stylesheet file that transforms ITM-generated XSML into hyperlinks. These hyperlinks are then added to the bottom of the HTML message body for Subscription Portal messages. By using a different XSL style sheet, the appearance of these hyperlinks can be changed.

6. Documents

When content sections in a service such as a subject or message body are added for a delivery method, a document is created. Documents define the fully-formatted content that end users receive. Documents combine static content such as text, HTML, and static images with dynamic content such as reports returned from MicroStrategy Intelligence Server, personalized files, and personalized Web hyperlinks. Multiple types of documents are available to support different types of content such as HTML, text, and Excel. Each delivery method, such as email or wireless, supports one or more content sections such as the subject or message body of an email message.

7. Subscription Sets

When defining a service, one of the primary questions that needs to be answered is "To whom will this service be delivered?" In Narrowcast Server, this question is answered by defining a subscription set for a service. A subscription is a request to deliver the content in a service to a recipient. A subscription also provides a particular address and personalization for this recipient. A subscription set is a collection of subscriptions that is used for the same service. Narrowcast Server has two fundamental types of subscription sets: static and dynamic. Static subscription sets store subscription information in the Narrowcast Server Subscription Book. Dynamic subscription sets execute an information object to retrieve user and subscription information from an external, dynamic data source such as a MicroStrategy project, a database query, or some other type of external system.

8. Schedules

A schedule sets the times or frequencies a service is executed, and represents a recurrence pattern, not a fixed date on which to send the service. For example, although you can be specific with the service start date (e.g., April 4, 2000, at 2:30 PM) generally the content will be sent on a recurring schedule such as a yearly recurrence. Execution intervals are calculated by combining the start date of a service with the interval specification of a schedule to arrive at a set of dates and times when the service runs.

9. Question Objects

   A question object is a container that holds a set of questions that users answer to define their preferences. It provides a template for users to specify their preferences by answering one or more questions. The answers to these questions allow information objects to tailor the information that is returned to each user.

10. Users and Addresses

    Users are the end users who receive Narrowcast Server services. Each user is given an "account," allowing a login ID and password to be associated with the user so that self-subscription interfaces can enforce authentication requirements. In addition, users allow all personalization settings to be defined at the user or "account" level. Finally, users provide the structure necessary to allow one user to have multiple addresses (such as a wireless PDA address and an email address) so that personalization can be applied consistently across communication endpoints.

    An address tells MicroStrategy Narrowcast Server how to send services to a particular subscriber. Each address can be associated with just one user, who may have multiple addresses. The address properties are as follows:

    - Address name
    - Physical address
    - Address display
    - Delivery method
    - Device
    - PIN
    - Status
    - Expiration date
    - Delivery window

    MicroStrategy Narrowcast Server supports many other advanced development paradigms including dynamic subscriptions, User Personalization, Supporting Diverse User populations and Page Personalization. Details on these may be found in the chapter dealing with Narrowcast Server architecture.

15.19 Customization / SDK Developer

The MicroStrategy BI platform is an open platform built upon standards that allows developers to customize the look and feel, and functionality of the platform.

Details of the customizability of the MicroStrategy Platform can be found in the chapter that details the Software Developer Kit (MicroStrategy SDK). The following section discusses some of the developer resources available to the Developer interested in customizing the look and feel of the MicroStrategy BI platform.

The MicroStrategy Developer Library

The MicroStrategy Developer Library contains over 6,000 pages of documentation and reference material, complete with API references, and how-to's for developers to quickly access information at the most detailed level.

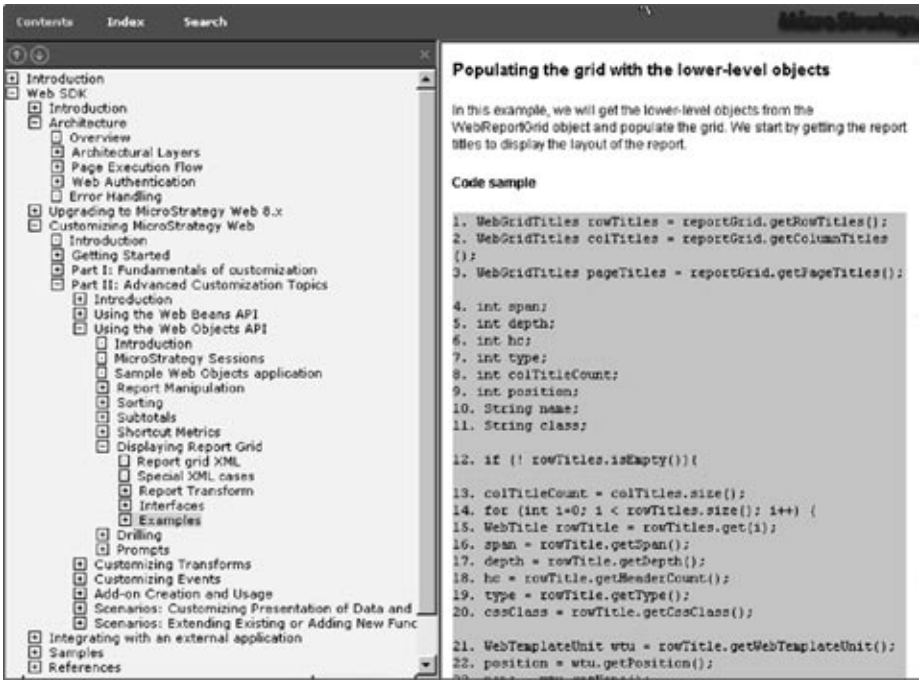


Figure 15.43 The MicroStrategy Developer Library is in a handy electronic format.

The MicroStrategy Developer Library provides:

- Comprehensive documentation detailing the MicroStrategy BI platform architecture
- API reference, complete with input/output parameters
- Over 70 task-oriented examples with sample code
- Tools and tips to upgrade to the latest version of the MicroStrategy BI platform
- Case studies for integration with external applications
- Sample applications and utilities

Type of API	Description
Web API	The Web API allows customization of the Web interface and integration with third-party applications. It enables modification of the look and feel of the Web interface, presentation of BI data and functionality available to end users.
Intelligence Server API	The Intelligence Server API exposes all the functionality of the MicroStrategy Intelligence Server, such as creating metadata objects, administering users and security, and tuning Intelligence Server parameters.
Narrowcast Server API	The Narrowcast Server API allows application developers to manage an information delivery application. This includes the ability to control the operations as well as develop new services. Narrowcast Server functionality can be extended further by adding new sources of information, data formatting capabilities for new devices, and data transmission capabilities.
Subscription Portal API	The Subscription Portal API comprises eight classes encapsulating the creating, editing or deleting of subscriptions, and their associated objects in a self-service BI application.
MicroStrategy Office API	The MicroStrategy Office API allows developers to build custom applications that use Microsoft Office products as a user interface to the MicroStrategy BI platform. These include connecting to a BI application, browsing folders and reports, specifying report parameters using prompts, and running and displaying reports and documents.

Figure 15.44 The MicroStrategy BI platform contains open APIs for all the major components.

15.20 Summary

The MicroStrategy BI platform is designed to give developers of all skill levels, from system architects to Web-based power users, powerful tools to build the most sophisticated BI applications.

With tools for translating data models into business models, architects can map different enterprise data sources into terms that are readily understood by business users. Powerful statistical and custom grouping tools allow developers to build advanced predictive models as well as advanced OLAP metrics, and groupings that can be used by other users. Web-based power users can then use these to create reports and formatted dashboards and scorecards. Developers can also build sophisticated alerting and batch reporting applications, and customize MicroStrategy end user interfaces through a comprehensive, well-documented API set.

16

Administrator Experience

As the scope of business intelligence systems expand with more data, more users and more applications, administration becomes a key driver in the cost of ownership. There are different administrative roles in an enterprise. Sometimes all these roles are taken on by just one person, but often the responsibilities are spread over several departments or teams.

16.1 In-depth Administration Across the Whole Platform

The administrators of business intelligence systems need to accomplish many tasks, including:

- Project Setup
  - Installing and configuring the BI system
  - Managing system performance
  - Controlling software licenses

- Access Control
  - Setting up users and their security profiles
  - Monitoring and enforcing security policies

- Operations Management
  - Ensuring data integrity after data loads
  - Maintaining a stable and reliable BI environment
  - Monitoring system usage to prevent performance bottlenecks
  - Troubleshooting user issues

- Project Lifecycle Management
  - Managing BI applications in development, test, and production environments
  - Assessing the impact of changes to the system
  - Identifying and decommissioning unused BI objects

Experienced system administrators evaluate the administrative capabilities of BI systems using two key criteria:

1. Visibility – can they access and track every key determinant of system performance, or are they forced to view the system as a "black box"?
2. Controllability – can they control every key parameter to tune the system to suit their own unique environment and to guarantee stability and performance, even under extreme conditions?

Superior software architectures are set apart by explicitly incorporating visibility and controllability features into the system. Visibility is delivered through detailed and granular statistics, error logging, and diagnostic files.

Controllability is achieved by providing a wide range of server and application tuning functions that let the administrator configure the system – both when first implementing the BI system and on an ongoing basis.

The architectural tenets for administering the MicroStrategy BI platform are:

- Allow completely centralized management and configuration of the business intelligence environment
- Allow fine-grained monitoring, logging, and control over the system's performance and use
- Provide fine-grained control over the types of operations various classes of users are allowed to perform
- Define the level of detailed diagnostic statistics to capture for troubleshooting purposes
- Provide real-time and historical analysis of system performance to identify trends and potential bottlenecks
- Provide impact assessment of proposed configuration changes and potential changes to the application definition
- Automate all repetitive tasks, and allow them to be initiated on time-based or event-based triggers
- Support bulk administration operations to simplify system changes, and reduce manpower investment

The MicroStrategy BI platform meets all of these requirements. MicroStrategy Intelligence Server, MicroStrategy Narrowcast Server and MicroStrategy Web all provide detailed administrative functionality, and MicroStrategy Administrator brings it all together for a BI system administrator.

Administrators overseeing a MicroStrategy BI system have complete visibility into, and control over system performance and tuning. This detailed visibility and control is provided in an easy-to-use fashion, and designed to be extremely powerful and time-effective.

This section covers the MicroStrategy platform's capabilities in each of the following areas:

- Real-time monitoring
- Error logging and diagnostic files
- Historical performance analysis
- Centralized administration
- System management software integration

16.2 Real-Time Monitoring and Control

At any point in time, the administrator can monitor system activity. MicroStrategy provides real-time information about important activities in the BI environment via the following administrator functions:

- Job Monitor – View currently executing jobs and jobs containing errors
- Project Monitor – View existing BI projects and their status
- User Connection Monitor – View active users and their session information
- Schedule Monitor – View a list of all scheduled reports and who scheduled them
- Database Connection Monitor – View connections to the metadata repository and data sources
- Cache Monitor – View report cache details and cache expiration dates
- Cluster Monitor – View clustered machines and their availability
- Scheduled Services – View the next time any report distribution service will be triggered
- Distribution Managers – View the currently executing report distribution services
- Execution Engines – View the execution progress of currently executing report distribution segments
- Debug Monitor – Receive real-time component-level statistics and debug information of the report distribution system

The administrator can also perform specific, related tasks directly from within these monitors, including:

- Discontinuing jobs – stops errant runaway jobs that take up system resources
- Maintaining projects – loads new projects, takes projects off-line, and adds projects to cluster nodes
- Disconnecting users – removes inactive users
- Disconnecting database connections – changes the number of concurrent database connections
- Expiring caches – invalidates old reports, and frees up memory
- Cancel Service executions – controls Narrowcast Server operations



Figure 16.1 Intelligence Server administration from the Desktop interface.

The Control Center allows real-time monitoring of a range of operating system and Intelligence Server performance counters. In addition, the administrator can also use the Windows Performance Monitor and the Windows Event Viewer or UNIX syslog facility for additional information about system performance. This integration is out-of-the-box, and no additional work is required. Over 50 performance counters are traced in the following functional areas:

- Current jobs executing in Intelligence Server
- Current users connected to Intelligence Server
- Process resource allocated to Intelligence Server

Real-time information for Narrowcast Server is viewed from the System Message Viewer in its administrative interface.

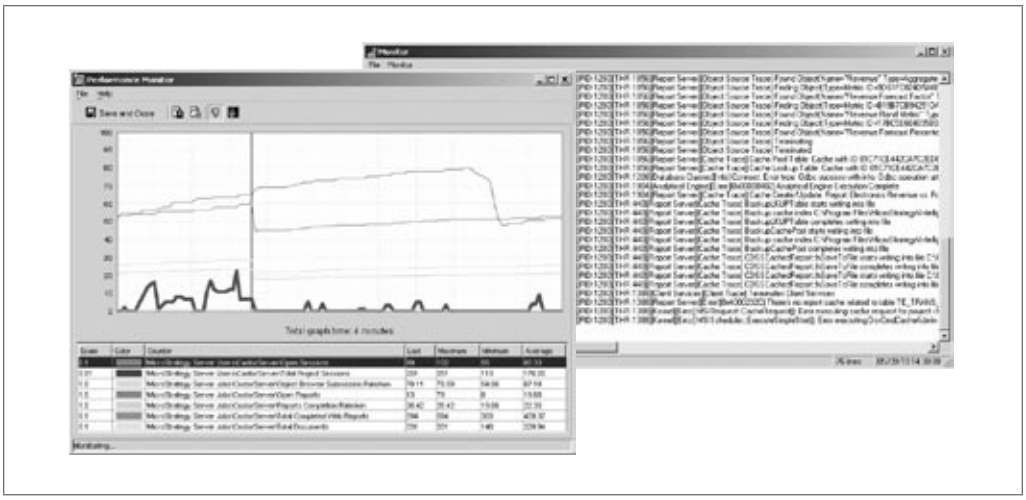


Figure 16.2 Monitoring the MicroStrategy BI environment

16.3 Error Logging and Diagnostic Files

Log files and operational statistics enable historical data to be accumulated so that cause and effect analysis as well as chronological analysis can occur. The MicroStrategy BI platform enables various types and levels of logging, with complete control over the granularity of data being accumulated. Multiple log files are available to let the administrator analyze system performance. The administrator can set the level of logging to record statistical data, errors, warnings or informational messages, depending on the amount of information needed.

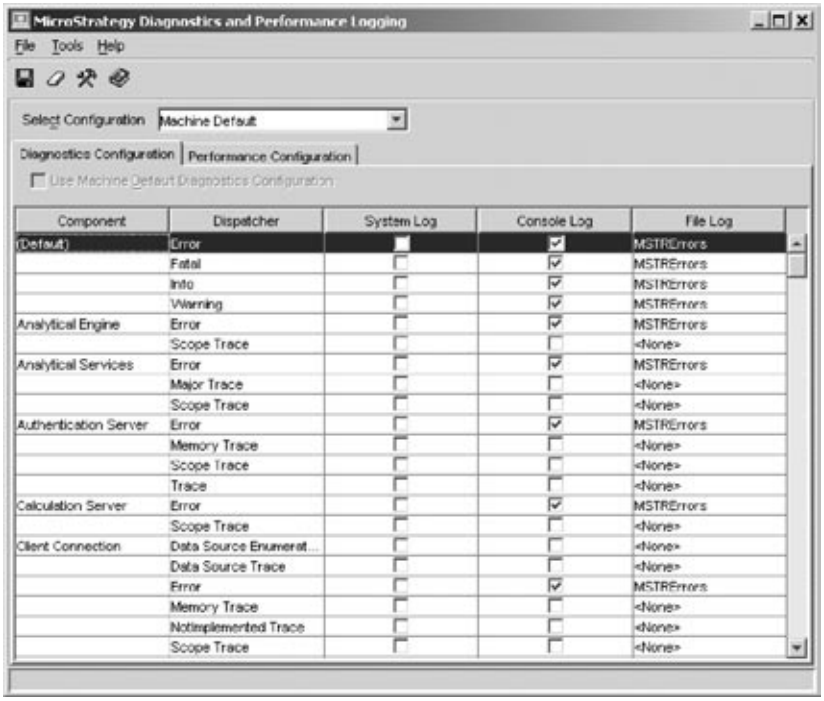


Figure 16.3 An administrator can conveniently control component level-logging from a centralized interface.

The administrator can view log file information from both the Windows Event Viewer and directly from flat files.

Administrators can use these logs for the following purposes:

- An observation tool to monitor current activity
- An error display console to identify and correct system faults
- A diagnostic tool to tune the reporting environment
- An aid that allows MicroStrategy Technical Support to help customers resolve issues in a timely and accurate manner

In extreme situations, log files can grow to a point where they impact operations. To avoid these situations in a production environment, a maximum size for log files can be specified. For easier management of log files, MicroStrategy uses the implementation of time or size-limited rotation strategies. A maximum number of log files can be specified to control the overall required disk space used for logging and diagnostics.

16.4 Enterprise Manager – Historical Performance Analysis

In addition to viewing current system status using monitors and error logs, MicroStrategy can store historical information on system usage. This information is stored in a statistics database. Leveraging this rich historical view of the system, temporary anomalies in usage and performance, which are of minimal importance, can quickly be distinguished from trends that may have a wider impact on the system. This type of long-term assessment is enabled through Enterprise Manager, one of the components of MicroStrategy Administrator.

Enterprise Manager lets system administrators analyze, and act upon detailed and granular logs produced by the MicroStrategy BI platform. It incorporates a complete data warehouse, analytical application, a data loading process, and dozens of insightful system performance and performance tuning reports.

Enterprise Manager uses MicroStrategy's own analytics to monitor and analyze usage statistics recorded by one or more Intelligence Servers. The Intelligence Servers capture user session and report usage statistics for each server, user, and reporting job. This information is written to a central database. The detailed information logged includes the servers, the users, the reports, and metadata objects, the time of submission, the execution time for various steps in report processing, and tables and columns that were accessed in the data warehouse.

Enterprise Manager provides over 80 reports categorized in four areas of analysis:

- Project analysis – Analyzes growth trends and displays summary and detailed information of the BI applications.
- Operational analysis – Analyzes user concurrency and project usage, and helps determine how system resources are being used by the MicroStrategy implementation.
- User analysis – Reports user connection activity and report execution history.
- Performance analysis – Reports detailed system performance trends, and provides insights into data warehouse tuning.

Many of the reports are parameterized to heighten flexibility, and speed up the analytical process. For example, some reports prompt for the time period to analyze (e.g., "today," "last week," and "last month").

Administrative reports and documents are accessible from all MicroStrategy interfaces, and are contained in a standard MicroStrategy project in a metadata repository. The Enterprise Manager project can be extended to incorporate additional data sources, and additional reports can be created to reflect unique characteristics of the environment. The reports help the administrator perform the following tasks:

- Analyze usage trends, and make decisions such as adding more hardware to the system
- Identify processing bottlenecks such as number of database threads
- Assess the effects of configuration changes and tuning strategies
- Identify the most active users and their time spent using the system
- Determine the best time to run scheduled reports in off-peak hours
- Identify the most popular reports, and determine which ones might need to be cached for optimum performance

Narrowcast Server can send Enterprise Manager reports and documents to administrators and project managers via email on a regular basis, providing an efficient mechanism for monitoring the MicroStrategy BI implementation.

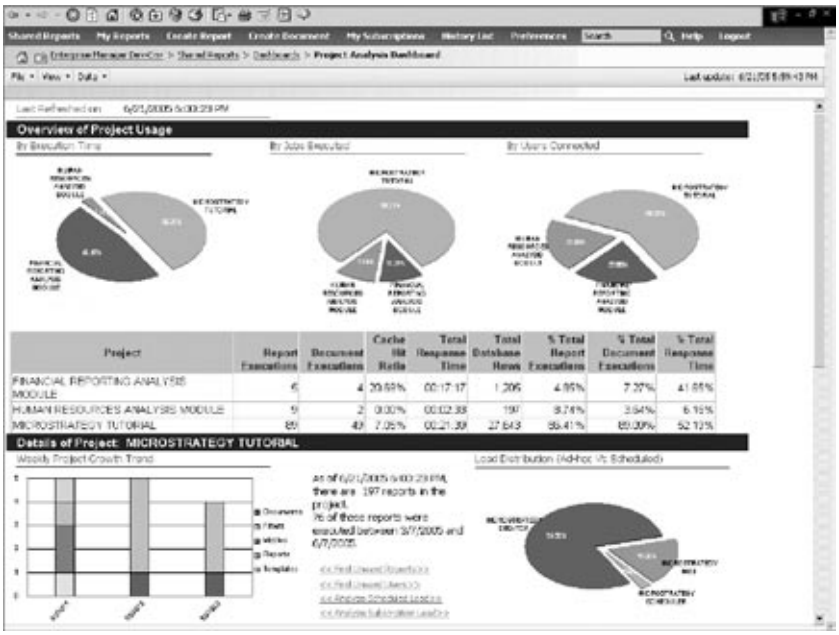


Figure 16.4 Project dashboard shows system usage at a glance in Enterprise Manager.

16.5 Centralized Administration

Identifying bottlenecks or errors using real-time monitoring, error logs, and trend analysis is the first step in enhancing a BI system's operational performance. Simple corrective action and granular control over the platform's operations is vital to ensure service level agreements. The MicroStrategy BI platform contains features, tools, and utilities that optimize, and manage all servers in the MicroStrategy environment from a single remote console.

Granular Parameters Provide In-depth Server Configuration

Intelligence Server configuration capabilities can be used to manage system resources, and to control user privileges.

1. System management and configuration – Each server can be configured individually to optimize the use of hardware resources by the server's components. Using the Desktop, an Administrator can:

    - Create connections from Intelligence Server to the databases
    - Add or delete servers from a server cluster
    - Expire or delete caches

- Manage job requests
- Set backup frequency
- Configure memory settings
- Create and trigger report schedules
- Configure query governing settings
- Define job thread prioritization
- Set the level of diagnostic reporting and logging

2. User and security profile management – User application privileges can be set for individual product features. Data access for any user can be customized down to the row and column level. Setting these controls is simple, and involves selecting check boxes or running pre-built scripts. No custom coding is required. Additionally, administrators can use graphical interfaces and text-based commands to:

    - Create and manage users
    - Define the user authentication mechanism
    - Organize users into groups and roles
    - Assign functionality privileges to users and groups
    - Assign data permissions to users and groups
    - Assign object access to users and groups
    - Manage user connections to the business intelligence projects

Information Governing Allows Real-Time Control of the Processing Environments

An important consideration in implementing and deploying business intelligence applications is managing the number of users, number of job requests, and amount of data that a server in the environment can process. In a MicroStrategy BI implementation, administrators can set limits to prevent servers from becoming overloaded by requests when extraordinary conditions exist. For instance, administrators can control the maximum number of users at any time. This capability ensures that the servers in the MicroStrategy environment run efficiently at all times.

To ensure that important reports and users are processed ahead of others, priority levels can be set according to the report, the user group, the application being accessed, or the interface being used. Items with higher priority are processed before lower priority items.

Governors can be applied at the server or application level. The list below shows many of the governor settings available in Intelligence Server.

- Server Level Governing
    - Maximum amount of memory used by Intelligence Server
    - Maximum amount of memory for in-memory report caches
    - Maximum number of job requests
    - Maximum number of user sessions
    - Maximum user session idle time
    - Maximum number of XML cells
    - Maximum number of XML drill paths
    - Maximum memory consumption for XML
    - Maximum memory consumption for PDF
    - Maximum memory consumption for Excel
    - Priority of reports based on job cost
    - Priority of reports based on user group

- Priority of reports based on project
- Priority of reports based on request type
- Priority of reports based on requesting application
- SQL time-out
- Maximum number of History list messages per user
- Number of days History list messages are active

- Application Level Governing
  - Maximum report execution time
  - Maximum number of report result rows retrieved
  - Maximum number of prompt elements retrieved
  - Maximum number of intermediate result rows retrieved
  - Maximum job requests per user
  - Maximum job requests per session
  - Maximum job requests per application
  - Maximum user sessions per application
  - SQL time-out per report

Effective Caching is Vital for Good System Performance

Caching speeds up operations by storing data and report results in faster accessible locations. A sound caching policy should be established that will maintain the integrity of all cached data – reports, prompt lists, and metadata objects. The MicroStrategy BI platform supports this in multiple ways.

- Monitoring – Detailed information about each cache file is available in real-time including creation time, status, type, size, the number of times accessed, the last time accessed, and expiration date. Enterprise Manager provides more analysis into the use of report caches, the cache hit ratio by report or user, and it identifies under-utilized caches.
- Managing Data Integrity – After data has been loaded into the data warehouse, all relevant cache files should be invalidated to ensure data integrity. This task can be automated by triggering time- or event-based schedules. For more granular control, cache invalidation can be tied to changes to data warehouse tables.
- Tuning – Cache creation for the most frequently accessed, and most important reports can be scheduled to run in off-peak hours. By monitoring the cache hit ratio, administrators can evaluate the effectiveness of their currently implemented caching strategy, and find the optimal balance between end-user performance and cost of added batch processing time.
- Governing – Intelligence Server maximizes caching efficiency and performance by storing the caches directly in memory. In order to ensure optimal use of server resources for all job processing, the amount of memory Intelligence Server dedicates for caching, and the total number of cache files it can handle can be specified.

Using any or all of these techniques, business intelligence applications will provide optimal performance to users while ensuring data integrity. Any project can be managed with several data load schedules providing near real-time business intelligence.

Command Manager Automates Standard Administrative Tasks

MicroStrategy Administrator eases the administrative workload through its Command Manager component. Command Manager provides scripting capabilities using text commands, bringing repeatability to the administrative process and featuring six core strengths:

- Scalability – Recurring administrative tasks that contain large volumes of changing data, such as user, security, and report cache management, is much quicker with command scripts than in an interactive graphical environment.
- Consistency – Often, commands in a script are essentially a repeat of the same pattern applied to a set of variables. Managing of hundreds or thousands of users with changing security roles and profiles is simplified using third-party tools, such as SQL, that can easily be used to generate a large number of similar commands. Also, scripts can flawlessly execute large numbers of commands in sequence.
- Efficiency – While commands duplicate the actions taken in a graphical user interface, running commands in a script removes the time between clicks. Repetitive tasks or generally large amounts of steps are significantly expedited. Asynchronous execution of scripts is fully supported.
- Reliability – Script files can be fully tested before being used in a production environment. Archiving script files have the added benefit of documenting the history of setup and maintenance activity in the BI system.
- Automation – Script files can be run automatically by a third-party utility or the operating system without human oversight. Critical and routine tasks can be scheduled to run during off-peak hours, freeing up resources during busy periods.
- Cost savings – Administrators can focus on higher level tasks instead of being tied up with routine tasks. Both small and large deployments of the MicroStrategy BI platform can be managed with minimal administrative overhead.

Key administrative functions are accessible and controllable through commands that can be run from a command line or a graphical interface. These commands can be saved in script files for reuse. Script development in the graphical user interface is simplified though the highlighting of the words in the command syntax. A wide variety of searchable script outlines relieves the burden of memorizing the command syntax.

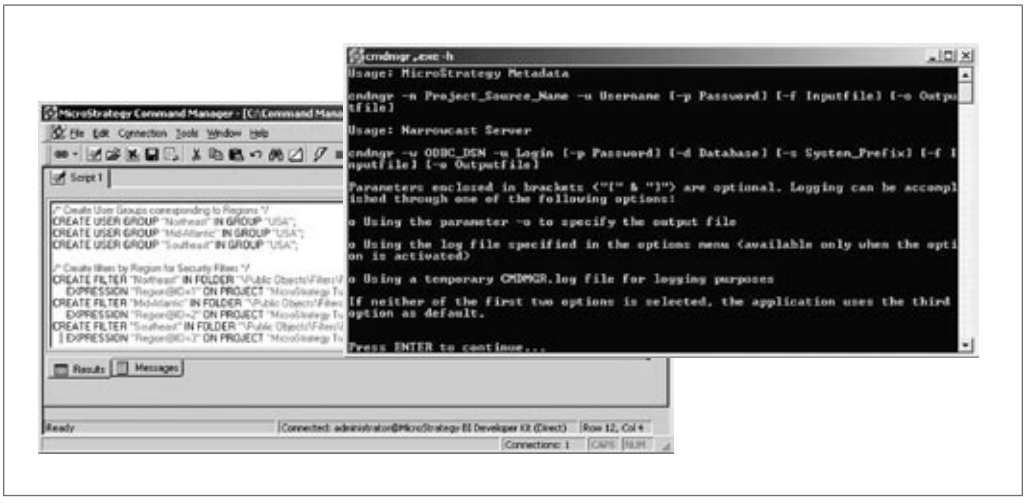


Figure 16.5 Command Manager runs commands from a graphical user interface or from the command line.

Command Manager contains over 200 commands that fall into the following categories:

- Security Management
  - Create and manage users and groups
  - Maintain security filters and roles
  - Grant and deny object access and privileges
  - Set up user profiles and connection maps

- Operations
  - Monitor jobs, users, and database connections
  - Set governing properties on projects and servers
  - Manage server clusters
  - Manage report and table-level caches
  - Trigger events, schedules, and report delivery services
  - Run operating system commands
  - Send messages to Windows Event Viewer

- Project Maintenance
  - Configure server and project parameters
  - Create and manage filters
  - Create and manage metrics
  - Create and manage attributes

Life-cycle Management is Crucial to Smooth Application Deployment

Best practices in application deployment suggest an environment where development, testing, and production environments are clearly defined and separated from each other. An application is allowed to progress to the next stage only when it has clearly met the objectives and requirements defined for a particular stage. To enable a smooth and quick transition from one stage to another, an enterprise-class business intelligence platform must contain a set of application development, and management tools that allow clean and easy migration of business abstraction objects from one stage to another.

MicroStrategy's administration tools can manage many reporting applications and information delivery systems. The reports and other metadata objects are first developed in a development environment, and tested under controlled conditions in a separate environment, before being deployed to the end users in the production environment.

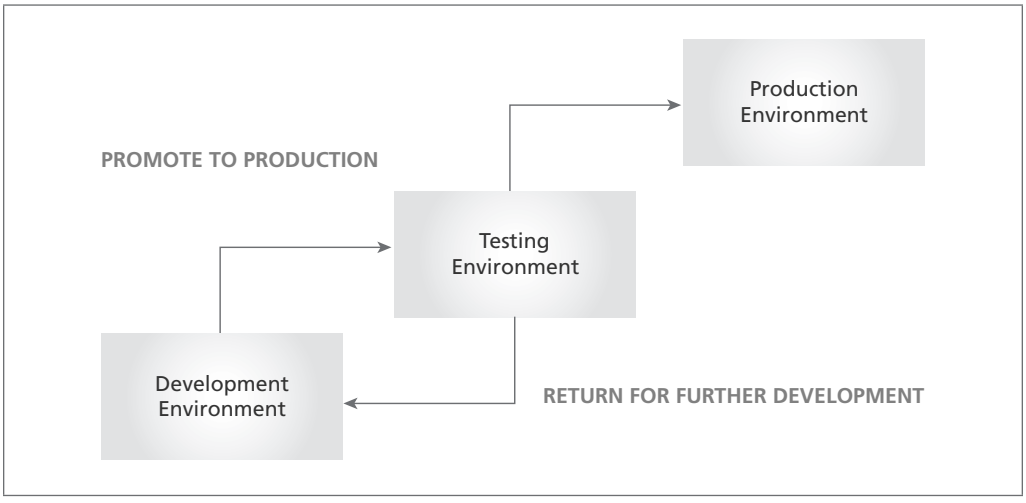


Figure 16.6 Migrating objects in the MicroStrategy BI platform.

The object management capabilities of the MicroStrategy BI platform were designed to support the migration of objects in various environments. The user interfaces make this process as easy as copying a file from one folder to another using a right-click menu or by dragging and dropping the objects. This simple task masks the complexity inherent in maintaining the integrity of every object in MicroStrategy's dynamic object-oriented architecture. Objects

depend on other objects to complete their own definitions. Reports depend on metrics, attributes, templates, and filters. Metrics depend on facts, functions, and other filters. Because of these dependencies, copying an object, such as a report, without also copying the objects on which it depends, would produce an invalid object in the destination or target system. The report would be unusable.

MicroStrategy automatically locates and migrates all of the dependent objects used by the object being migrated, ensuring the integrity of objects in all environments. When a report is migrated from one environment to another, any templates, filters, attributes, and metrics used by that report are also migrated. Since each of these objects can also have further object dependencies, MicroStrategy recursively ensures that objects used by those objects are migrated as well, until all required objects are copied.

Object Manager Facilitates Change Management

Object Manager is a graphical interface used to interactively manage metadata objects in related projects. Through a single, searchable console with drag-and-drop functionality, metadata objects such as facts, attributes, hierarchies, metrics, filters, reports, and other reporting objects can be migrated, copied or deleted quickly and accurately. Object access permissions can also be set in Object Manager.

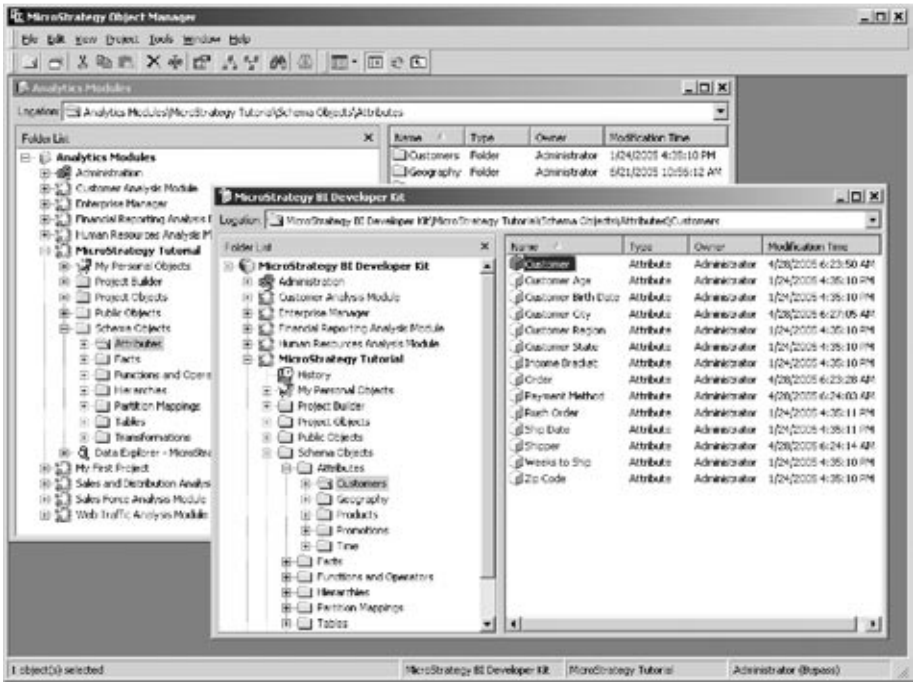


Figure 16.7 Object Manager simplifies moving objects between projects.

Understanding the effect of changes to metadata objects is vital to managing BI applications effectively. Object Manager determines the impact of modifications made to any MicroStrategy object by displaying parent and child objects. For example, if the definition of a metric has changed, an administrator can easily view the parent objects (reports, documents, templates, filters, and other metrics) and child objects (facts, filters, tables, and columns) of that metric with a click of a button, showing all objects affected by changing the metric. With this information, all affected users can be notified before the change is made. Impact analysis is also critical for documenting changes to BI applications, and accurately estimating the resources required to implement changes and upgrades.

The following tasks are available in the Object Manager and Narrowcast Server Administrator interface:

- Manage report objects
- Share report objects
- Copy projects and make backups
- Compare projects in different environments
- Promote objects from development to test to production
- Search for objects based on name, type, owner, and time of last update
- Search for unused objects
- List parent and child object dependencies

Upgrade BI Applications with the Project Merge Utility

When upgrading a production project with application changes from a development environment, project managers do not want to rely on manual drag and drop operations; instead, they prefer an efficient automated process.

MicroStrategy provides this ability with the Project Merge utility, a command-line utility that automates the simultaneous migration of all project changes. Conflicts can be resolved through the configuration of rules that are stored in an XML file, and reused with each merge. The set of rules are defined through the Project Merge Wizard in four different levels:

- Object Category
- Object Type
- Folder
- Object

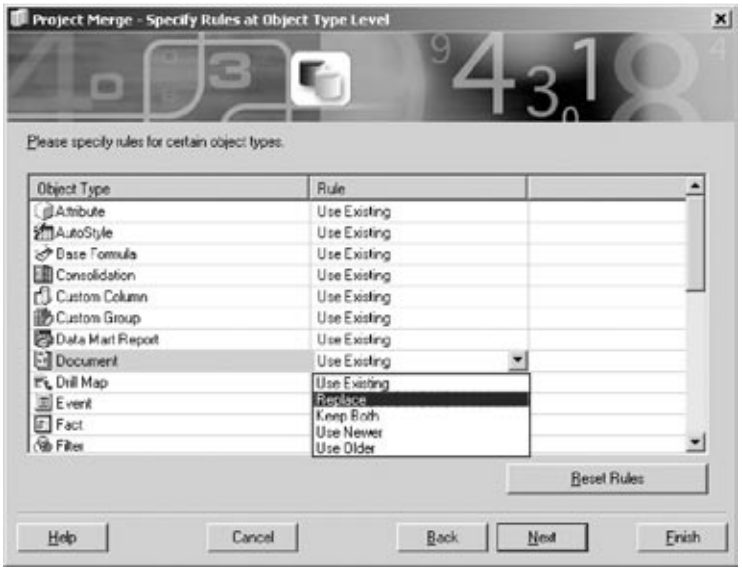


Figure 16.8 Specifying upgrade rules by object type in the Project Merge utility.

Tracking Software License Deployment

The MicroStrategy BI platform helps to keep track of all deployed software licenses. An automatic daily scan aggregates the user privileges and reports if software licenses are being exceeded. The scan can also be run manually using the License Manager utility, which also displays details about the software installed on the machine, and the licenses allocated for users. Using the License Manager, one can:

- View system information and the version, edition, expiration, installation history, and other details of the products installed on the machine
- Determine the product licenses in use by a specified user group or user
- Print the license usage information or export it in HTML format.
- Upgrade licenses by providing the new license key, without reinstalling the products
- Activate or deactivate a MicroStrategy server installation

16.6 System Management Software Integration

Many IT organizations implement software management applications that monitor the complete hardware and software environment. These products allow system exception handling to occur automatically, and in accordance with predetermined rules, guaranteeing predictable behavior even when a human administrator is unavailable.

Using logging databases, diagnostic files, and Windows Event Viewer, third-party system management software such as IBM® Tivoli®, CA UniCenter®, HP Openview®, and BMC® Patrol® can seamlessly integrate with the MicroStrategy BI platform. Automated, real-time tuning of the MicroStrategy's servers ensures the best possible performance and throughput. Two key MicroStrategy platform features make this possible.

1. Obtaining real-time component level information and statistics about MicroStrategy operations programmatically
2. Controlling each function of the system in a programmatic fashion through Command Manager commands or directly through MicroStrategy APIs

For example, if the system management software detects that the load on the data warehouse is reaching a critical level, the number of connections opened by the Intelligence Server can be reduced automatically to alleviate the load. System management software can control Intelligence Server at a granular level, and perform the following tasks:

- Start, stop or idle Intelligence Server
- Change query governing parameters
- Change the number of open database connections
- Switch to another Intelligence Server if the original server stops responding
- Load a project that experiences peak traffic on additional nodes in a cluster
- Cancel jobs that are causing bottlenecks in the data warehouse

These, and other administrative capabilities provide the ability to automate Intelligence Server configuration changes, thus increasing system availability, improving performance and report throughput, and reducing system administrator workload.

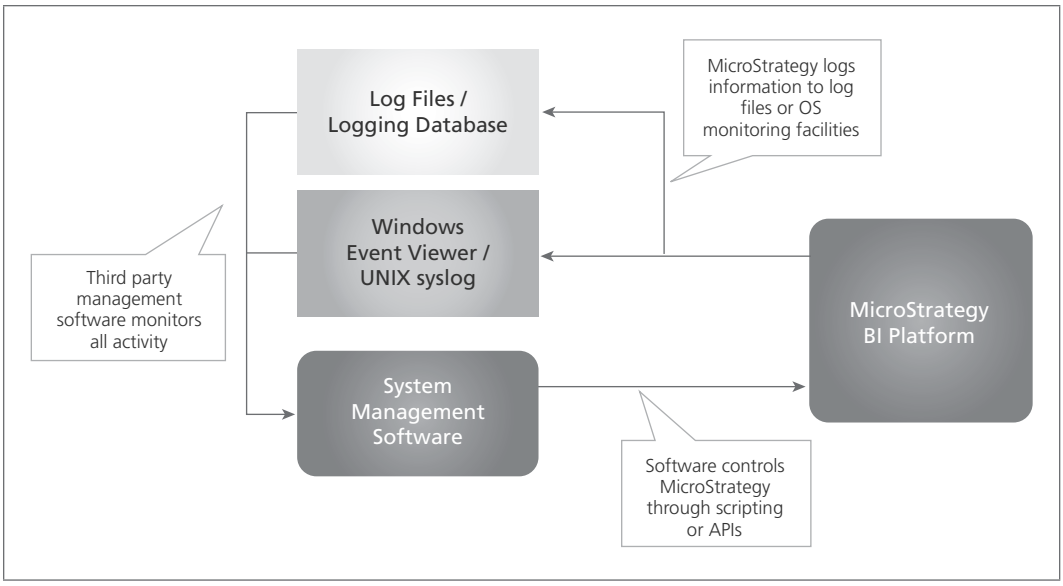


Figure 16.9 Integration with third-party system management software.

The diagram above illustrates how the MicroStrategy BI platform integrates with system management software. Intelligence Server and Narrowcast Server periodically log state information to the logging database, NT Event Monitor, or to diagnostic files. The system management software monitors activities in these log files, identifying situations that require action. When the system management software detects one of these situations, it adjusts the configuration of the Intelligence Server or Narrowcast Server accordingly using Command Manager commands or open API calls for those products. System management software can greatly automate the system administrators' activities, and is one more way that MicroStrategy reduces the total cost of ownership.

16.7 Summary

The administrative capabilities of the MicroStrategy BI platform provide the most comprehensive system management environment in the business intelligence industry. MicroStrategy is the only BI platform in the industry that ensures 99.999% uptime of business intelligence applications providing access to all corporate data to all corporate users. It contains a powerful set of tools that offer a comprehensive environment for developing, deploying, monitoring, and maintaining small-, medium-, and enterprise-scale systems.

IMPLEMENTING BI APPLICATIONS

17

Developing Business Intelligence Applications

Over the years, MicroStrategy's thousands of customers have implemented tens of thousands of BI applications. Consequently, MicroStrategy has designed the MicroStrategy BI platform to make implementation easier, and speed up deployment. Customer methodologies have ranged from loosely defined preferences to the most detailed and strictly enforced implementation policies.

Regardless of the nature of the implementation, there are common themes in the application development and implementation lifecycle. This chapter begins with a brief discussion of the common BI application lifecycle based upon the tens of thousands of BI applications built using the MicroStrategy BI platform.

Irrespective of an organization's starting point with BI – whether they have departmental solutions that expose a single style of BI or highly functional enterprise-wide applications that enable all 5 styles of BI – there is a natural evolution of successful BI applications. Evolution is driven by:

- Increasing number of users
- Increasing amounts of data
- Addition of new data sources
- Introduction of new application requirements

Only MicroStrategy provides an industrial-strength BI technology that is suited for all levels of BI deployments – from departmental, to divisional, to enterprise-wide and extranet deployments. These deployments are covered following the discussion of the application lifecycle. This chapter ends with a brief discussion of the various factors that influence capacity estimation for BI deployments.

17.1 Application Lifecycle

The typical BI implementation involves four key stages in the application development and implementation lifecycle:

- Plan
- Develop
- Deploy
- Maintain

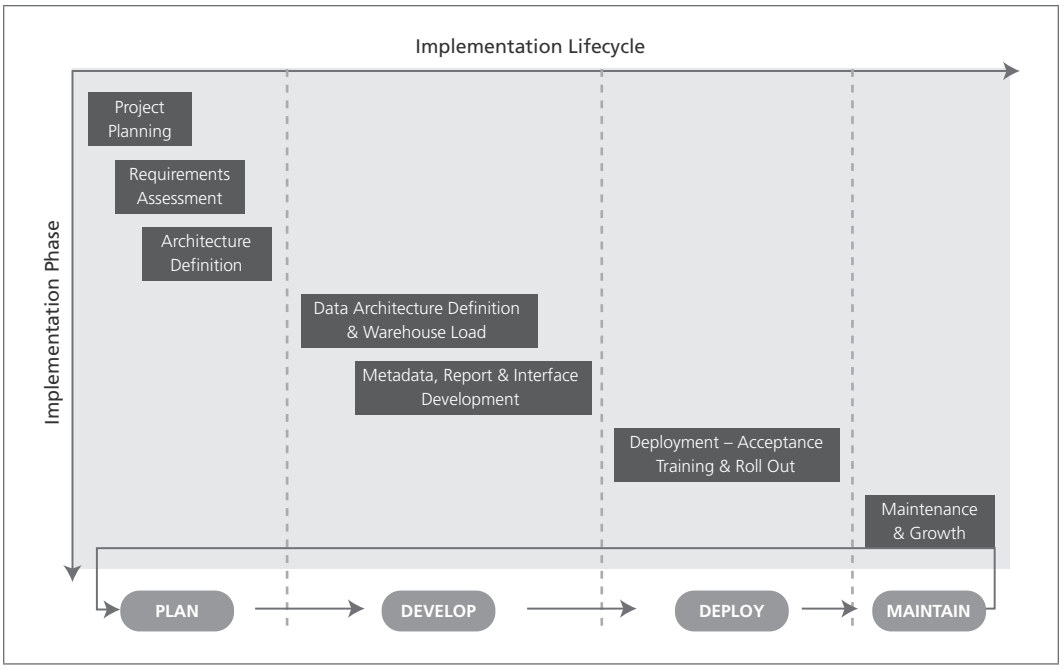


Figure 17.1 The high–level project plan of a typical BI implementation lifecycle.

Plan Stage

The Plan stage produces a comprehensive project plan and application architecture for the implementation of the BI application that serves the business needs. In general, this stage involves four types of activities:

- Project Definition
  Project definition begins with a conceptual vision for a solution, and defines the steps, costs, staffing, and risk mitigation strategies required to implement the conceptual vision. During this stage, project planners define the functional scope, phases of implementation, and acceptance criteria for each phase of the implementation. Project definition also includes education and training plans for developers as well as end-users of the application.

- Requirements Gathering
  Requirements gathering involves identification of specific business needs served by the BI application. It identifies functionality that the future application needs to offer in the following categories:

  - Reporting, analysis, and monitoring
  - User interface(s)
  - Users and user training needs
  - Data structures, data movement, and transformation
  - Infrastructure: hardware, software, networks, and security
  - Integration with other applications

- Architecture Definition
  Architecture definition defines the plan for augmenting the existing technical architecture to support the new application. This requires the assessment of architectural constraints, infrastructure capacity, system software standards, hardware standards, and development standards. Architecture definition also results in initial capacity plans for the development, testing, acceptance, training, and production environments. Planners

develop the technical architecture based upon the system integration requirements, initial capacity plans, and functional requirements of the application. This includes factors such as number of end-users, their application requirements, planned usage, and service level requirements. The typical deliverables of the Architecture definition include:

- Hardware and software architecture for development, test, acceptance, training, production, and maintenance steps of the implementation cycle
- Security architecture
- Backup and recovery strategy
- User Interface Style guide

Develop Stage

During the Develop stage, project teams build the physical application solution scoped in the planning stage. It is composed of the following primary activity groups:

- Design

  The design step translates the highly specific business requirements obtained from the requirements gathering step into actionable specifications. The output of the design step includes detailed specifications on each category of application requirements. In some cases, organizations develop a prototype application before conducting full-scale development.

- Construct

  In this step, the development team builds all components of the BI application. This involves building the data warehouse, extract-transform-and-load routines, schema layer, application layer, and reports, scorecards and dashboards. All activities around developing custom user interfaces or extending MicroStrategy functionality occur in this step.

- Test

  The test phase involves testing the application at both the unit level as well as the system level. Unit tests ensure that individual components of the application match the specifications, while system tests ensure that all components of the BI application work as designed.

  In addition, organizations conduct stress or volume testing to ensure that the system can withstand the application load expected in the production environment. The stress testing allows organizations to identify performance bottlenecks, which can be addressed using numerous tuning settings available in the MicroStrategy BI platform, and through modifications to database, ETL tools, application servers, Web servers, network, and system management tools.

Deploy Stage

The Deploy stage integrates the newly developed application into the organization. The typical steps in the Deploy stage include:

- System Acceptance

  In system acceptance testing, a set of users trained on the application goals, system usage, and success criteria approve the application for rollout to the rest of the target audience. Typically, acceptance testing occurs on a mirror of the production environment.

- Training
  One of the most crucial, but often overlooked steps in application deployments is the requirement to train the user base, including users and application administrators.

- Rollout
  During the rollout step, organizations deploy the new application to users, and shut down the legacy system.

Maintain Stage

The Maintain stage includes activities that ensure that the BI application and underlying infrastructure continue to meet business expectations. These activities include:

- Providing a responsive Help Desk to address business user questions
- Conducting regular system upgrades
- Performing periodic system and database monitoring, tuning, and administration

The success of the BI implementation spurs additional development requirements due to:

- Growth of user community
- New reporting and functionality requirements
- Data growth and the addition of new data sources

The Maintain stage closes the loop on application development by feeding vital information for improvements back to project sponsors, planners, and application developers.

17.2 Evolve from Departmental to Enterprise BI Applications

Organizations are in various phases of adoption of Business Intelligence. Companies that are embarking on BI initiatives for the first time typically limit themselves to departmental BI solutions. The MicroStrategy architecture's plug-and-play components make it a perfect choice for small departmental applications. A company can only use the scorecarding capabilities of MicroStrategy, if all they need from the application are scorecards for executives. Customers who only need email alerting can deploy MicroStrategy's alerting functionality. Invariably, companies that start with narrowly focused BI applications expand those applications with new styles of BI over time. MicroStrategy's integrated architecture makes it easy to "plug in" new components while enhancing or maintaining existing ones.

This ability to future proof BI applications to scale to large user populations, or to increase analytic scope, allows application designers to roll out new functionality incrementally. No matter how organizations begin their MicroStrategy implementations, they can be confident about their ability to increase the number or functional richness of their BI applications, the number of users who access those applications or the amount of data these applications access.

Departmental Deployments

Companies often start using MicroStrategy for a single, departmental BI application. MicroStrategy's ease of use, and low administrative overhead make it a perfect choice for these initial departmental applications.

The characteristics of departmental deployments include:

- A short development cycle that leverages existing data models and schema
- The provision of any style of BI, with the option of adding functionality upon user request. For example,

organizations start with Enterprise Reporting and effortlessly add OLAP slice-and-dice capabilities as needed.

- Quick deployment to 10s through 100s of users through an interface of their choice – Web browser, Windows Desktop, Microsoft Office suite, email, printers, and file servers
- Seamless scalability to account for user and data growth

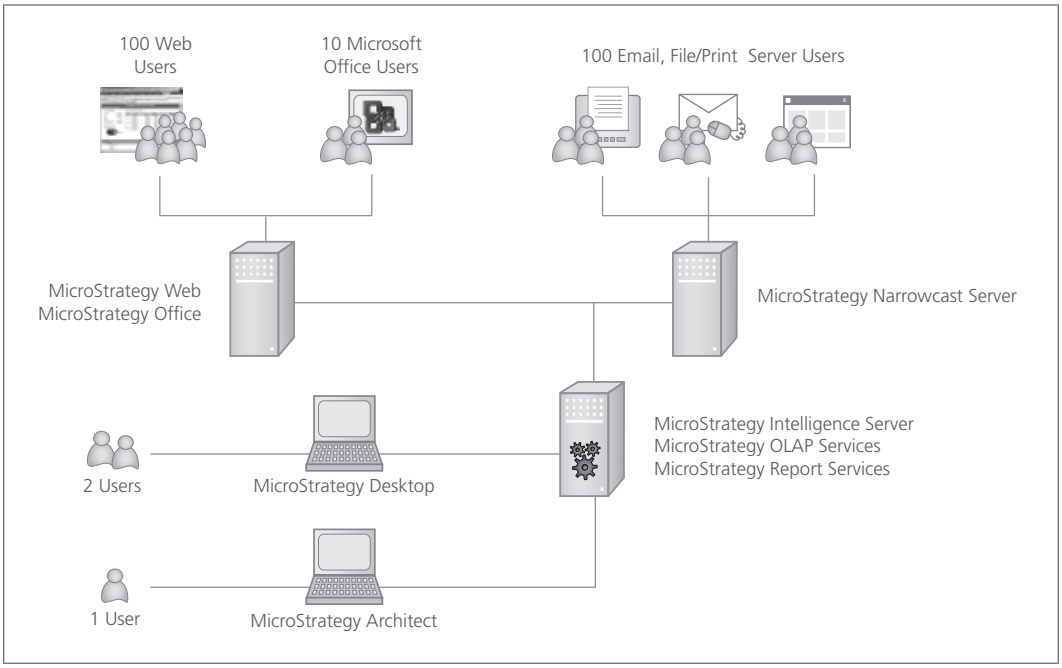


Figure 17.2 An example of a departmental deployment of the MicroStrategy BI platform.

With departmental deployments, it is common to install MicroStrategy Intelligence Server, MicroStrategy Report Services, and MicroStrategy OLAP Services on a single computer server. Modularity of the MicroStrategy BI platform ensures that Report Services and OLAP Services are only installed if pixel-perfect reporting and OLAP Analysis are required by users. Separate servers host MicroStrategy Web and MicroStrategy Narrowcast Server. A handful of users manage the complete application development and deployment cycle.

Divisional Deployments

As a company's BI appetite grows, it typically demands additional styles of BI. For example, a company may have started with an operational reporting application, and now needs to implement scorecards for its executives. Companies need support for larger user groups and data volumes, with ever-increasing requirements for security.

The characteristics of a division-level BI deployment include:

- 1 to 5 Applications hosted on the same BI infrastructure
- Applications use all 5 styles of BI
- Rapid deployment to thousands of business users across the whole spectrum of user interfaces supported by the MicroStrategy BI platform
- End user self service and powerful report delivery to email, printers, and file servers
- Single, robust security model for all applications and interfaces
- Comprehensive administration and change management infrastructure
- Clustered Intelligence Servers, Web Servers, and Narrowcast Servers for fault tolerance and reliability

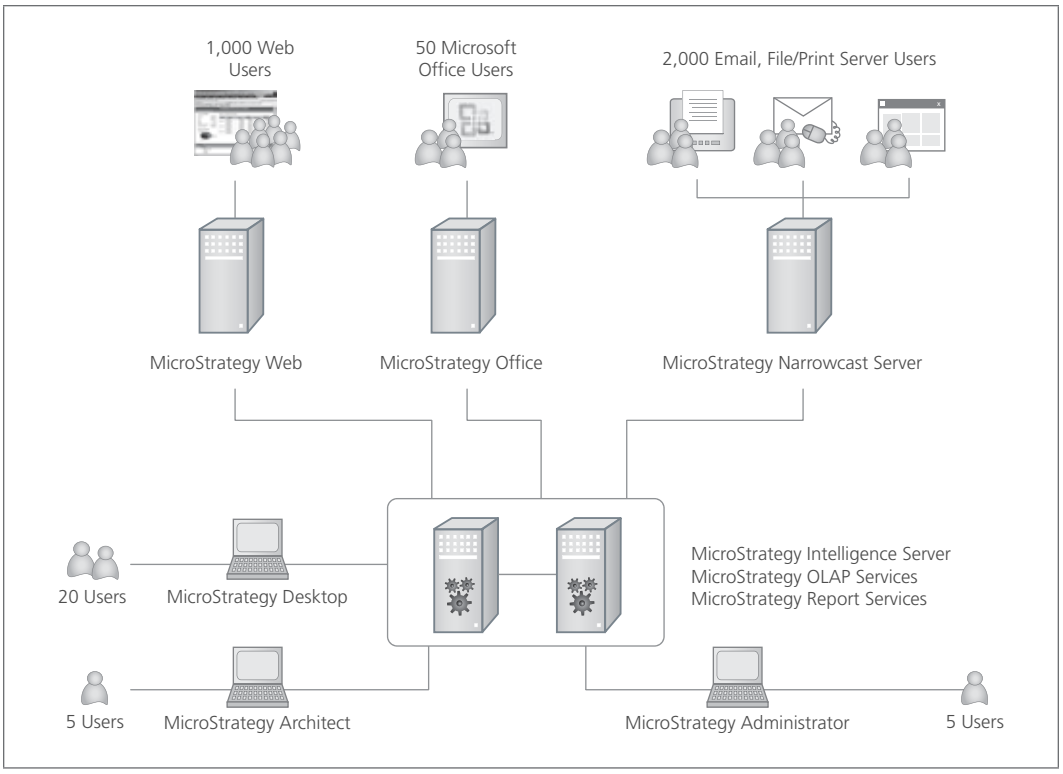


Figure 17.3 An example of a divisional deployment of the MicroStrategy BI platform.

With divisional deployments, it is common to separate the development environment from the test and production environments to improve control over the development process. The development environment includes some BI architects, and a small set of report developers and interface designers. In many cases, the BI architects and report developers are the same person. In addition, a handful of administrators manage application migration from development, to test, to production.

Enterprise-wide Deployments

As BI becomes an enterprise utility, usable by all business users, companies can easily scale their MicroStrategy implementation by adding additional styles of BI, by increasing capacity for tens of thousands of users, and by adding infrastructure to support many applications. Since MicroStrategy's dynamic metadata objects are reusable, objects such as metrics, reports, and security definitions can be reused in new applications.

Other BI technologies require each developer to create their own report components (e.g., metrics, filters, custom groups, and drill paths) for each report without relying on reusable components, compounding the effort of developing new applications. This approach places the burden of enforcing consistency across all reports on the developer. While this design philosophy delivers rapid results initially, the manual effort expended in making new reports consistent with previous reports adds considerably to the development effort. When this manual consistency checking fails, conflicting reports impede accurate decision-making.

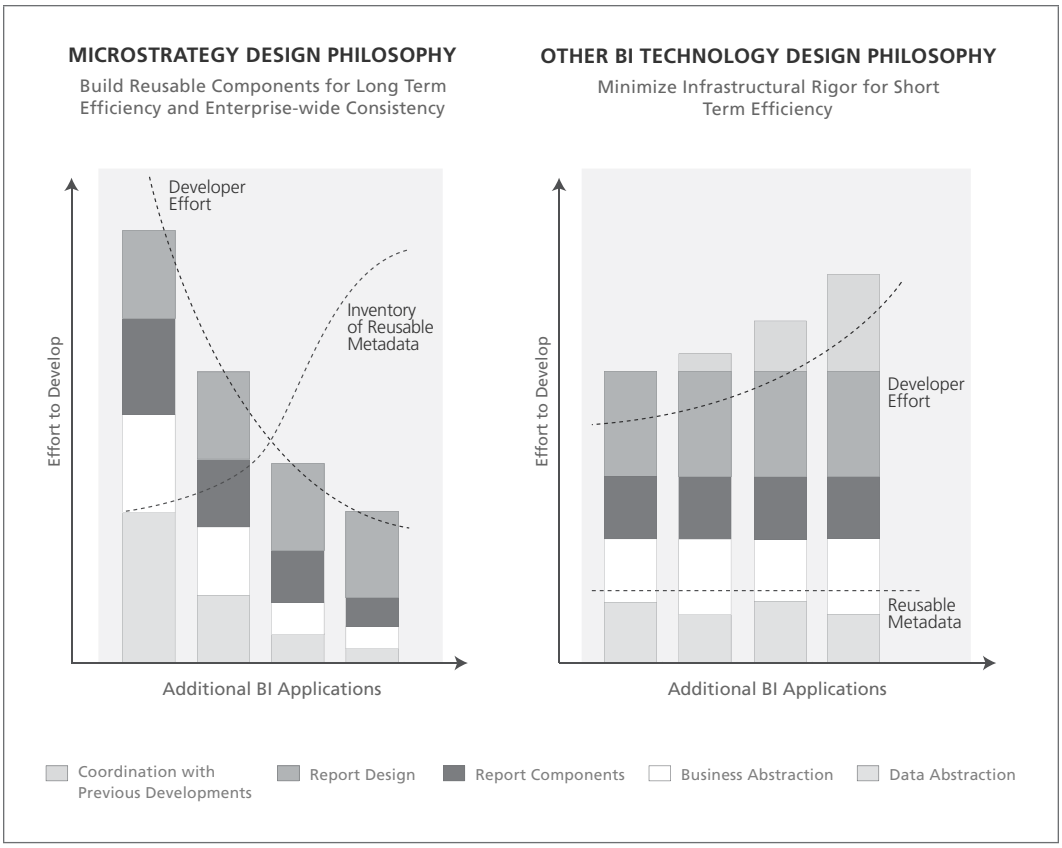


Figure 17.4 MicroStrategy is optimized for long-term efficiency, with each report developed more quickly than the last because it builds on previous development efforts.

In addition to the deployment characteristics of divisional deployments, the hallmarks of enterprise-wide deployments include:

- Over 20 applications hosted on the same BI infrastructure
- Very large user communities of over tens of thousands of users
- Distributed development teams
- Enterprise caliber security for all applications and interfaces, including single sign-on authentication
- Comprehensive change management, monitoring, and administration of the BI application
- Deployment through Enterprise portals and integration with external applications using Web Services
- Effortless capacity increases through the addition of more servers that host Intelligence Server, MicroStrategy Web, and Narrowcast Server

The MicroStrategy architecture minimizes administrative costs by automating critical but often mundane administrative tasks such as managing user privileges, purging caches, and expiring jobs. In addition to a graphical user interface or GUI-based administration, MicroStrategy gives administrators over 200 predefined scripts that automate tasks.

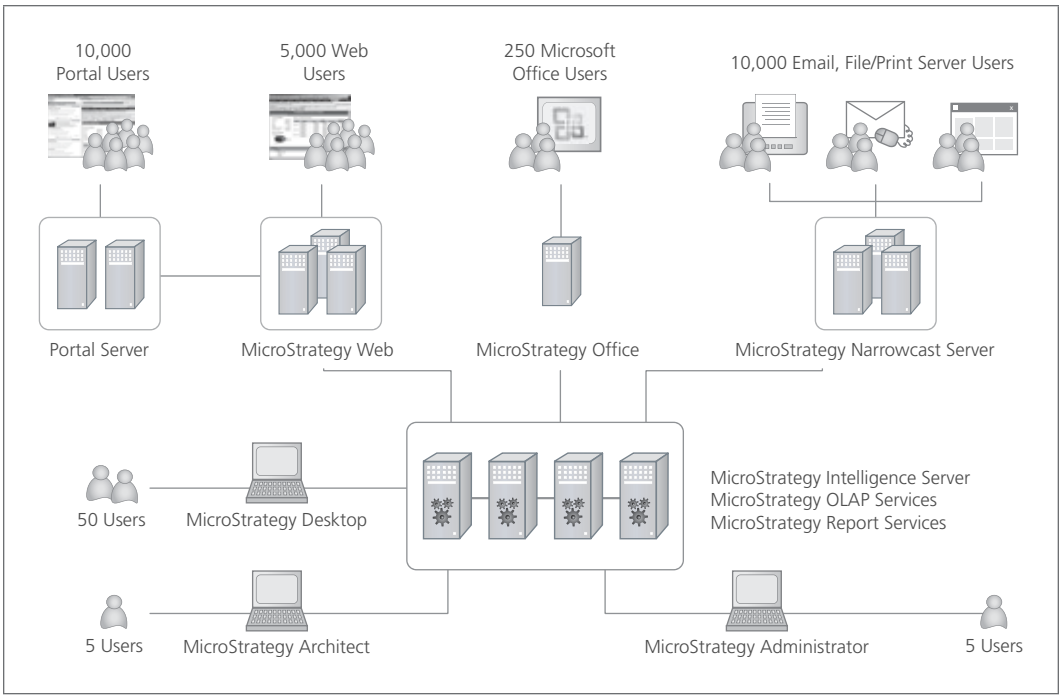


Figure 17.5 A typical configuration of MicroStrategy products for enterprise-wide deployment.

Extranet Deployments

With the success of BI applications in improving corporate performance, organizations are looking to implement BI applications to analyze and monitor external operations. Extending an existing application or building new applications to reach suppliers and customers places additional challenges on applications due to:

- Large distances between the business users and application infrastructure
- Increased security requirements
- Large numbers of external users
- Greater administrative overhead due to user management and enhanced system management

The geographic spread of users, and the resulting placement of the various MicroStrategy products for optimal performance are of particular interest during deployments. The remainder of this section discusses key considerations with respect to the placement of the various MicroStrategy components – Data, Intelligence Server & Narrowcast Server, Web, application server, and client.

- Placement of Report Servers and Metadata Repository
  Frequent interaction of Intelligence Server, Narrowcast Server, and the metadata repository mandate close proximity of the servers to the metadata. A dedicated high speed, high bandwidth network greatly increases performance due to reduced network latency.

- Placement of Intelligence Server and Data Warehouse
  The same reasons cited for the relative placement of Intelligence Server and the metadata repository apply to the placement of the data warehouse. While the Intelligence Server and data warehouse can be spread out geographically, a dedicated high speed, high bandwidth network greatly increases performance.

- Placement of Intelligence Server and Narrowcast Server, Development, and Administrative clients
  Due to the high throughput XML communication requirements of the Intelligence Server and Narrowcast Server, typical BI implementations ensure close proximity of Intelligence Server and Narrowcast Server. A dedicated high speed, high bandwidth network reduces performance problems. Similarly, a dedicated network is recommended between the development/administration interfaces (MicroStrategy Architect, MicroStrategy Desktop, and MicroStrategy Administrator) and Intelligence Server.

- Placement of Intelligence Server and Web Servers
  While a high bandwidth network is desirable between Intelligence Server and the Web Servers, many performance-enhancing strategies implemented in the MicroStrategy platform, such as multi-level caching, incremental fetch, and client-side processing of user actions, reduce traffic between the Intelligence Server and Web Server.

- Placement of Web Servers relative to Web users
  In general, of the four tiers of an extranet BI deployment, the network bandwidth between the Web Server and Web browser is the most unreliable and least controllable by the organization. For this reason, when users are highly distributed and high bandwidth networks do not exist, it is common to place Web Servers closer to remote users.

The MicroStrategy architecture provides numerous features that address which have been discussed in great depth throughout this book:

- Scalability for large user deployments
- Scalability for data growth
- Superior monitoring and administration of the BI system

17.3 Estimating the Capacity Requirements of the BI Application

A critical step in the planning and development of a BI implementation is estimating the capacity requirements. This task approximates the initial computing resources needed to meet an application's objectives and service level agreements. Capacity estimation is achieved through either an informal or a structured approach, which fall into the following categories:

- Following generic guidelines
- Using rules of thumb
- Comparing information from similar systems and workloads
- Extrapolating from unit tests
- Extrapolating from capacity models based upon a series of unit and system tests
- Performing benchmarks that simulate the actual load

Although a more structured methodology produces more accurate results, it also uses more resources and time, and therefore comes at a much higher cost.

Managing user requests, retrieving data from data sources, adding analytical richness, formatting and distributing report results, and performing report interactions all use computing resources. Increases in the number of users, the number of BI applications, the volume of data, and the amount of analytical reporting handled by the BI servers

also increases the amount of computing resources needed to maintain similar levels of performance and report throughput. Capacity estimation gives a reasonable idea of the amount of computing resources needed to power a BI implementation. The main system requirements that should be estimated are:

- Processing power
  Processing power is generally the most critical resource as it is the most expensive and hard to change investment. This involves determining the number of physical computer servers, and the speed, bit size, and number of CPUs.

- System memory
  The memory requirements of the BI application can influence the choice of the operating system and, in turn, the processing power decision. For example, any application process running on a 32-bit Windows platform can use only 2–3 GB of system memory. In contrast, 64-bit Windows, UNIX, and Linux operating systems support a theoretical 18 exabyte address space for application use.

- Hard disk space
  In terms of computing resources, hard disk resources represent the cheapest of the three types of system resources and are also the least critical.

Capacity estimation should be an ongoing exercise, rather than a one-time endeavor, to ensure that the system infrastructure continues to support a successful BI application deployment. Assumptions and factors must be regularly refined with data from ongoing monitoring and tuning activities. While there are numerous factors that affect the capacity requirements of a BI application, the most important factors include:

- Number of users
- Design of reports
- Interactivity of reports
- Monitoring and administration

Capacity Estimation Factors	Affect Processors?	Affect Memory?	Affect Disk Space?
Number of Users			
Total Users			✓
Active Users		✓	
Concurrent Users	✓	✓	
Reports Design			
Report Size		✓	✓
Analytical Complexity	✓	✓	
Report Layout	✓	✓	
Report Interactions			
Interactive Web Reporting	✓	✓	
Batch Reporting & Delivery	✓	✓	✓
Exporting Report Data	✓	✓	
Implementation and Administration			
Metadata		✓	✓
Caching		✓	✓
Monitoring	✓	✓	✓

Figure 17.6 Capacity estimation factors and their influence on processor, memory, and hard disk resources.

Estimating the Number of Users

The total number of users of an application is the easiest factor to estimate. Since user information is stored in the metadata and subscription book repositories, increasing the number of users increases the hard disk requirements. Depending on how they use the application, users have a secondary impact on the CPU and memory requirements.

At any given time, only 10-15% of all users are actively using a typical BI application. Intelligence Server allocates memory and address space for each active user from a shared pool of system resources. The larger the number of active users connected to a BI application, the more memory resources are consumed by Intelligence Server.

Typically, only 10-30% of the users logged into an application run jobs on Intelligence Server simultaneously. Thus, BI applications have a user concurrency ratio of 1-5% of the total named users of the application. Concurrent users directly influence the processing and memory resources of the computer servers.

Understanding Report Design Characteristics

The number of reports in a BI application is an important factor in determining the capacity of BI applications. However, the number of reports alone does not provide an accurate estimation; the size, analytical complexity, and report layouts must also be considered.

- Report Size
  Report instances are stored in a shared pool of memory called the working set. The amount of memory used depends on the amount of data and data types used, and this can be estimated relatively accurately.

- Analytical Complexity
  MicroStrategy adds analytical richness to reports by cross-tabbing data, adding calculations, and subtotals, and managing prompts. Reports that require a large amount of analytical processing require more processing power and memory, but the exact amount can be difficult to quantify. Existing implementations with similar characteristics provide a better estimate.

- Report Layout
  MicroStrategy formats reports in many ways, such as cross-tab grids, graphs, HTML, PDF, CSV, and Excel. Applying formats to report results uses both processing power and memory. The more datasets, formatting properties, controls, and report objects contained on a report, the more processing and memory resources are consumed.

Incorporating the Effects of Report Interactions

The MicroStrategy BI platform supports all 5 styles of BI, and provides the full spectrum of user interactivity within BI applications. These interactions affect different parts of the BI implementation, and can be categorized as follows:

- Interactive Web Reporting
  Most MicroStrategy BI applications are deployed on a tiered Web infrastructure. This provides a very rich reporting, analysis, and monitoring environment that is centrally maintained. MicroStrategy contains a broad range of functionality, from basic activities such as accessing the system, finding, running, and printing reports, to powerful analytical features such as drilling, sorting, pivoting, adding calculations, and exporting. Sophisticated report development capabilities are also available in the platform.

  The quantity and variety of report interactions consume both processing power and memory on all tiers on the Web infrastructure – Intelligence Server, MicroStrategy Web, the data sources, and the Web browser. Capacity should be estimated for all of the tiers.

- Batch Reporting and Delivery

  Many organizations process commonly used reports and scheduled reports in batches. Ideally this occurs during off-peak hours and periods of low system usage. The key constraints that govern system resources for batch reporting are the quantity of reports executed and delivered, and the size of the batch window in which the processing occurs. Shorter batch windows and more reports require more processing and memory resources on Intelligence Server and Narrowcast Server to complete report creation and delivery in the allotted time.

- Exporting Report Data

  Report data can be exported to many different output types, such as Microsoft Excel, HTML, PDF, and CSV. The main items that govern resource consumption when exporting are the output type, amount of data, formatting of the data, frequency of export, and user concurrency.

The processing power and memory consumption on MicroStrategy Web and Intelligence Server increase with an increase in any of these items. Large data volumes also affect resource consumption on end user machines running Web browsers.

Administering and Monitoring BI Implementations

The MicroStrategy BI platform provides the richest application metadata, with numerous monitoring and performance enhancing features to ensure successful implementations. Increasing functional richness and continued performance-monitoring take up system resources that must be included during capacity estimation.

- Metadata

  The central metadata repository in the MicroStrategy platform can house many BI applications. The size of the metadata has an influence on the memory consumption of Intelligence Server. Upon startup, Intelligence Server loads the server and metadata configuration information, and the data abstraction objects for each application, into memory. Intelligence Server's memory footprint increases when loading more applications or when loading large applications.

- Caching

  The MicroStrategy BI platform provides a comprehensive, multi-level caching architecture to improve throughput, optimize query performance, and reduce end user response times. Intelligence Server caches are stored both in memory and on disk. Caching affects not only the memory footprint of the Intelligence Server, but also the allotted hard disk space required to store caches off-loaded to disk. The cache monitor in Desktop provides accurate information about the number and size of caches in existing implementations.

  In addition to shared caches, personal reports can be saved to individual users' history lists. These reports increase the amount of hard disk capacity and memory needed on Intelligence Server.

- Monitoring

  MicroStrategy logs statistics, errors, and counters that monitor processor resources, memory usage, system throughput, and user activities. Storing the usage and performance statistics consumes a negligible amount of processor and memory resources, but additional hard disk space is required to store the log files. The capacity estimation exercise must account for performance monitoring and usage statistics logging. It is important to note that these monitoring resources should be used to improve future capacity estimation and tuning activities.

17.4 Summary

The MicroStrategy BI platform provides all the ingredients of an industrial-strength BI technology ideally suited for all levels of BI deployments – from departmental, to divisional, to enterprise-wide and extranet deployments. Irrespective of where organizations begin with their BI implementations – whether they have departmental solutions that expose a single style of BI or highly functional enterprise-wide applications that enable all 5 styles of BI – MicroStrategy deployments grow with the organization to meet the evolving requirements of:

- Increasing number of users
- Increasing amounts of data
- Addition of new data sources
- Introduction of new application requirements

With thousands of customers and tens of thousands of successful BI applications under its belt, MicroStrategy has designed the MicroStrategy BI platform to make implementation easier, and speed up deployment. Customers worldwide ensure their BI success and mitigate deployment risk by relying on a team of experienced MicroStrategy Professional Services experts. The Services team leverages decades of cumulative industry, process, and technology-specific knowledge, and provide best practice recommendations to ensure the success of BI applications.

18

Portable Analytic Modules

Every day, companies around the world in all major industries rely on MicroStrategy to give their entire enterprises a leading edge in corporate performance management (CPM). Executives, managers, and business analysts yield tremendous benefits from the ability to report, analyze, and monitor business processes with BI applications that measure the performance of the corporation as a whole, departments within the corporation, individual business processes deployed, and customer and supplier interactions.

With over 15 years of experience in business intelligence, MicroStrategy has developed a CPM starter kit, called the MicroStrategy BI Developer Kit, which bundles application modules, development tools, and a development framework and methodology. Like packaged applications, the BI Developer Kit enables rapid deployment of BI applications. Like custom applications, the BI Developer Kit allows companies to reuse their data warehouse, and adapt easily to changing business requirements. The MicroStrategy BI Developer Kit gives organizations the best of both worlds: packaged applications that can be customized easily to fit precise business requirements, and the development environment to do just that.

18.1 Rapid Application Development Framework

The MicroStrategy BI Developer Kit is a rapid application development framework that contains the following components:

1. Analytic Modules
    i. Prepackaged metadata – complete metadata repository of best-practices reports, dashboards, key performance indicators, attributes, metrics, filters, and custom groups
    ii. Default physical and logical data model – analytics that work off of a logical data model that can easily be mapped to an existing data warehouse or packaged data warehouse schema
    iii. Reference Guides – documentation on the data model, analysis areas, metadata object definitions, data dictionary, and report usage scenarios
    iv. The analytic modules span the following key business areas:
        - Customer Analysis
        - Financial Reporting Analysis
        - Human Resources Analysis
        - Sales And Distribution Analysis
        - Sales Force Analysis
        - Web Traffic Analysis

2. Implementation Methodology
    i. Implementation and Portability Guide – step-by-step guide for implementing analytic modules against existing data warehouses
    ii. Best Practices for Building Analytic Modules – design rules for designing and developing analytical applications

3. MicroStrategy BI Developer Products
    i. MicroStrategy Desktop Designer – development tool to create, modify, extend, and deploy analytic applications across the enterprise
    ii. MicroStrategy Architect – development tool that allows mapping of analytic modules to existing data warehouses

Each analytic module contains a set of reports, key performance indicators (KPIs) and business metrics, and sample scorecards. Developers and users use these reports and key performance indicators as building blocks to create and deploy additional reports and performance measurements that are specific to their organizational needs.

MicroStrategy provides report-building wizards to design and build reports. Developers also have at their disposal over 270 functions and operators (arithmetic, aggregate, statistical, financial, mathematical, OLAP, and data mining) with which to build any new KPIs and business metrics.

For example, a developer can use the revenue metric included in the customer analytics module to build additional metrics such as the maximum, minimum, running sum, moving average, standard deviation, future value, and seasonal forecasts of the revenue. Other areas of analysis and navigation paths can be customized, and new workflow wizards let end users build their own reports to analyze information. Customization that traditionally took months can now be achieved in days.

18.2 Portable Analytic Applications

Packaged analytic applications have had limited success in the marketplace because they are difficult to implement and customize, and they require large investments of time and resources. A better approach, known as portability, is to build packaged reporting modules that are independent of the physical data warehouse schema using an abstraction layer that maps the logical business structures to the physical database.

MicroStrategy portable analytic applications offer the benefits of packaged applications without the constricting requirement of conforming to a vendor-provided data-model and schema, or the need to modify existing warehouse structures to work with vendor-provided reports and analyses. Businesses can now leverage their existing data warehouses without the need for additional data extraction, transformation, and loading processes.

Developing a portable analytic application is different from developing a typical business intelligence application. The portability paradigm is governed by a set of rules and premises that should be adhered to when designing and building these applications. These rules span the three main development phases:

1. Analytic Domain Definition
   Adherence to standard analytic domains ensures the application applies to a wide number of warehouse schemas, and that it leverages standard processes and best practices for the analytic domain. Portable applications for analytic domains should represent one of the following:

- Standard business process – e.g. sales force automation, human resource management
- Business problem area – e.g. customer retention, product affinity
- Functional area – e.g. marketing, finance
- Standard data source – e.g. Web logs, sales and distribution data

2. Multi-dimensional Model Design
   The multi-dimensional model is a key component of the portable applications architecture. MicroStrategy portable analytic modules use an abstraction layer between packaged reports and the underlying data warehouse. It is imperative that the multi-dimensional data model complies with the following rules:

   - Modular Architecture – minimizes the impact of missing data attributes from the data warehouse, and makes implementation methodical
   - Standard Analytic Modules – focuses on horizontal functional areas, and makes the applications applicable in many vertical industries
   - Simple Hierarchical Structures – reduces the dependence on the underlying physical schema and complex parent-child relationships
   - Physical Schema Independence – ensures applications are not restricted by underlying physical structures, and works with all data warehouse schemas

3. Report organization and development
   This important step involves creating the final package, including the documentation of the reporting components. Report design and development in portable analytic applications should be conducted according to the following rules:

   - Organized Report Library – reports grouped by distinct analysis areas, dictated by key reports and metrics
   - Non-Database Specific Analysis – defines the business logic in the abstraction layer, and allows portability across database platforms
   - Schema-Independent Abstraction – uses underlying definitions to build key performance indicators and reports that ensure changes to definitions permeate seamlessly throughout the application
   - Package Standard Reports – focuses on standard best practices reports to provide a starter kit for analytic applications that can easily be implemented and extended

The portable analytic applications in the MicroStrategy BI Developer Kit are built on a complete platform architecture that makes customizations, extensions, and deployment an easy and manageable task.

18.3 MicroStrategy BI Developer Kit Analytic Modules

The Analytic Modules in the BI Developer Kit are a set of packaged analytic components, including reports, scorecards, KPIs, and business metrics, which are intended as starter kits to begin developing custom analytic applications specific to each organization. The modules are designed on top of a pre-defined data warehouse schema but can be mapped directly to existing data warehouses. The packaged best practices in terms of reports and metrics reduce application development time significantly, making this a low-risk, high return investment.

Customer Analysis

The Customer Analysis Module includes over 40 reports and 65 key performance indicators and business metrics for marketing and customer relationship management. The components help segment customers, monitor lifetime customer value, improve customer satisfaction, and drive profitability. The Customer Analysis Module's best practice analytics are fully documented in a 160-plus page reference guide.

1. Key Areas of Analysis
    - Acquisition, Retention and Attrition
    - Customer Segmentation
    - Profitability and Cross-Sell Analysis

2. Key Questions Answered
    - Has the rate of attrition, retention, and acquisition changed over time?
    - What is the impact of tenure on customer retention and attrition?
    - Are valuable customers more likely to leave?
    - What is the revenue loss attributable to customer attrition?
    - What is the distribution of customers by demographic and psychographic characteristics, and by lifetime value and revenue contribution?
    - How have demographic and psychographic characteristics changed over time?
    - What percent of revenues are being contributed by a given segment of customers?
    - What are the revenue, profit and margin contributions of customers by customer profile?
    - What are the characteristics of the most profitable customers, and what products do they buy?
    - Which products have increasing sales momentum?
    - Which products sell well together?

3. Key Attributes
    - Customer Geography, Customer Demographics, Lifetime Value Score, Customer Psychographics, Time, Revenue Deciles, Profit Deciles, Product, Customer, Affinity Product, Customer Status

4. Key Performance Indicators and Business Metrics
    - Active Customers, Lost Customers, New Customers, Retained Customers, Attrition Rate, Retention Rate, Attachment Rate, Revenue, Revenue Per Customer, Profit, Margin, Profit Variance, Revenue Variance

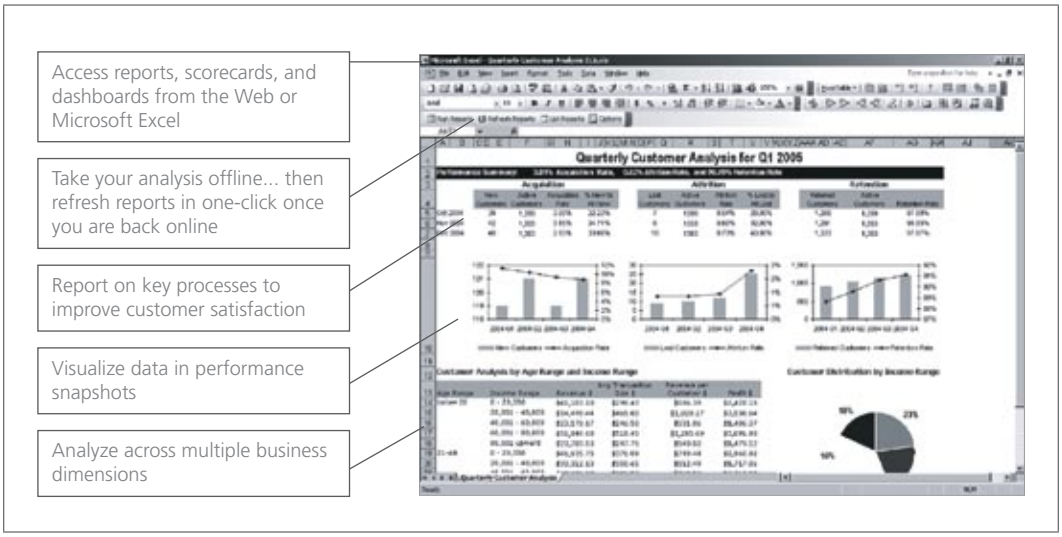


Figure 18.1 Customer-centric organizations benefit from reporting, analysis, and monitoring with MicroStrategy, yielding higher customer loyalty and satisfaction. As shown above, business users may opt to run MicroStrategy reports from Microsoft Office Excel, PowerPoint, Word, or Outlook.

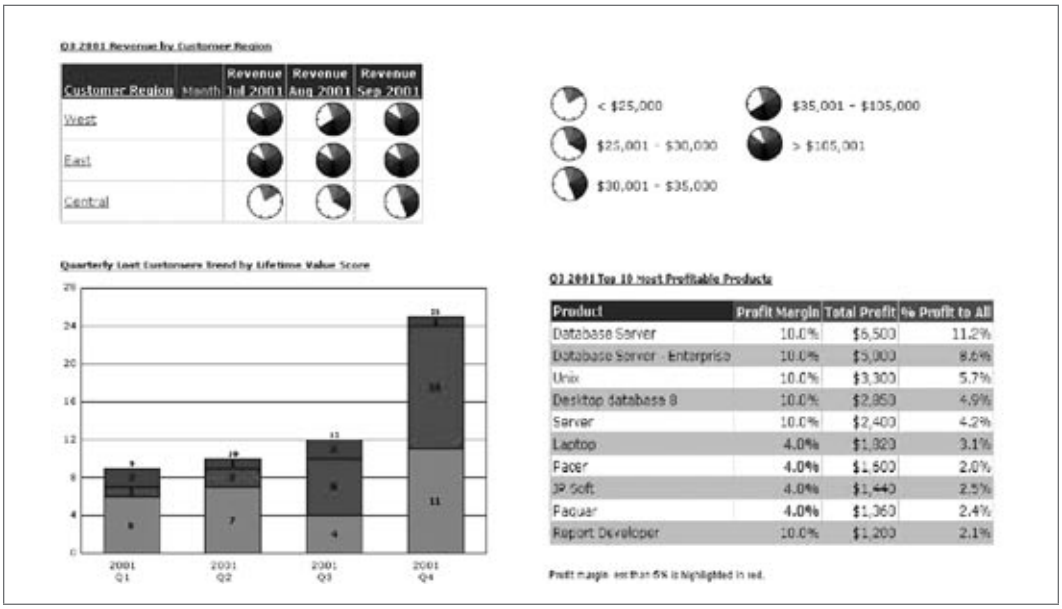


Figure 18.2 An example of a Customer Analysis Scorecard.

Financial Analysis

The Financial Analysis Module includes over 40 reports and 50 key performance indicators and business metrics used to obtain in-depth knowledge of financial information to increase corporate transparency and ensure Sarbanes-Oxley compliance. Its best practice analytics are fully documented in a 200-plus page reference guide.

1. Key Areas of Analysis
    - Accounts Payable
    - Accounts Receivable
    - Balance Sheet
    - Cash Flow
    - Costs and Expenses
    - Revenue and Forecasts
    - Profit and Loss Statements
    - Planning and Forecasting

2. Key Questions Answered
    - What is the breakdown of costs by vendors, and what are the associated trends?
    - Which bills are due this week, and for what amounts?
    - What is the aging distribution of Accounts Payable and Accounts Receivable?
    - Are there any customers with payment problems? If so, who needs to be notified?
    - What is the value of assets, liabilities and owners' equity on a given date?
    - What is the change in cash position period over period?
    - What are the revenue trends for the business units?
    - What are the trends in revenue-by-revenue types, if applicable?
    - What is the breakdown of expenses by business unit?
    - What is the forecasted revenue? Has this forecast changed, and what caused the change?
    - What is the Actual Amount profit margin by business unit or region? What are the associated trends?
    - Which business units are hitting their targets?

3 Key Attributes

- Account Type, Account Class, Vendor, Customer, Invoice, Time, Employee, Business Unit, Corporation

4. Key Performance Indicators and Business Metrics

- Accounts Receivable Aging, Accounts Payable Aging, Actual Amount, Forecast Amount, Planned Amount, Variance, Open Amount, Operating Income, Write-off Amount, Net Income

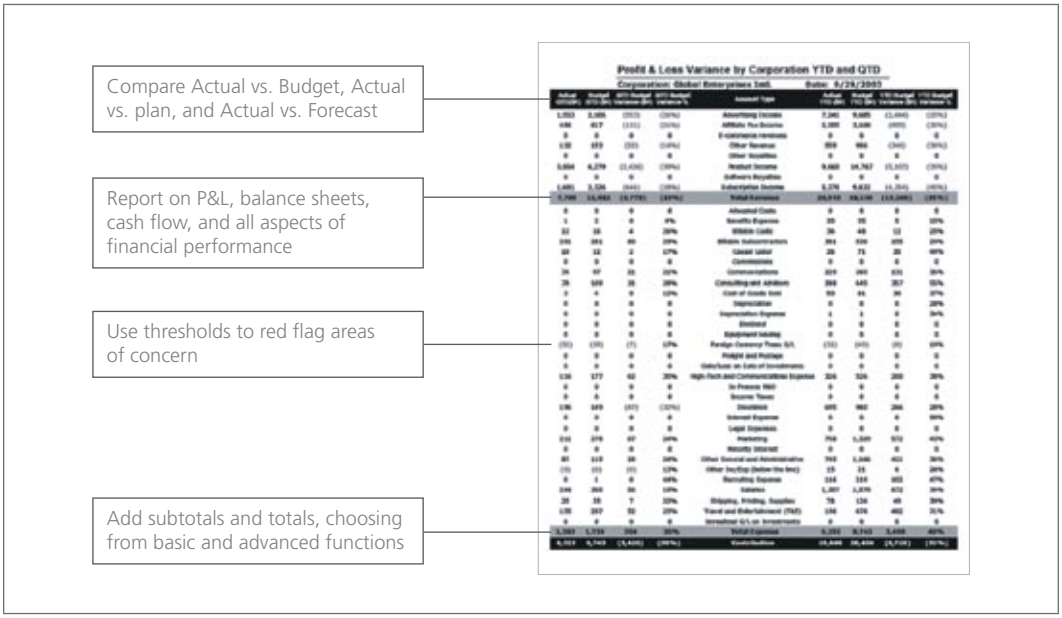


Figure 18.3 Financial reports in MicroStrategy meet exact printing and executive specifications, and allow for full investigative analysis.

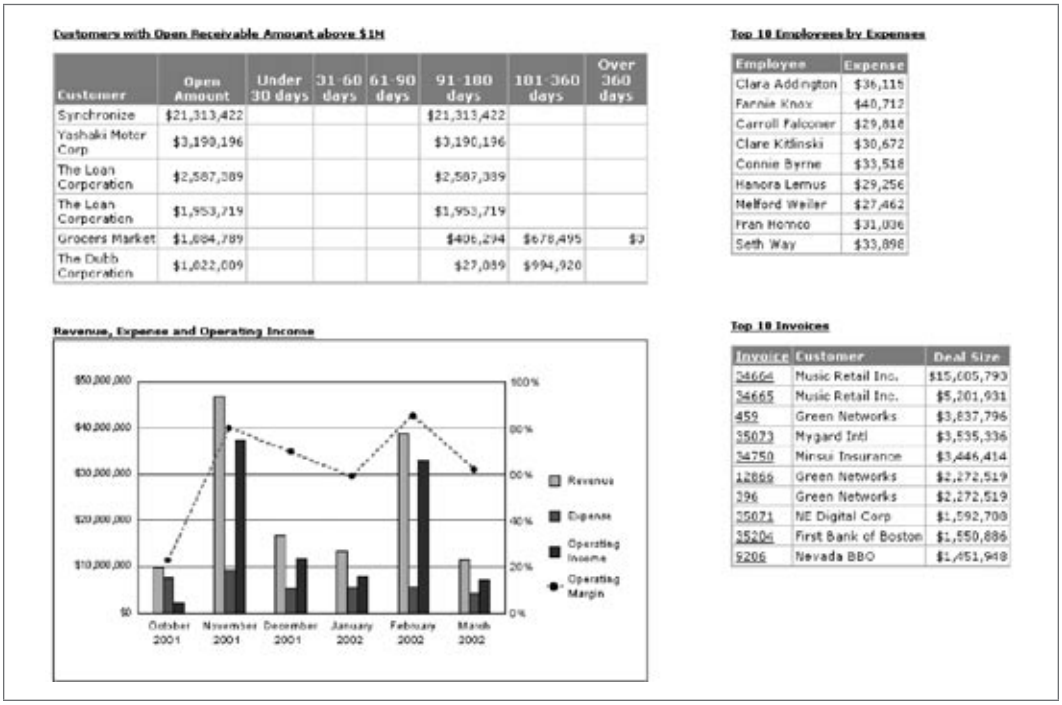


Figure 18.4 Sample Financial Analytics Report.

Human Resource Analysis

The Human Resources Analysis Module includes over 45 reports and 80 key performance indicators and business metrics designed to help your HR professionals monitor employment and labor law compliance, optimize the tremendous fixed costs associated with human resources, and improve employee recruitment, retention, and performance. Its best practice analytics are fully documented in a 250-plus page reference guide.

1. Key Areas of Analysis
   - Workforce Analysis
   - Compensation Analysis
   - Benefit Analysis
   - Attrition Analysis
   - Recruitment Analysis
   - Workforce Development and Productivity

2. Key Questions Answered
   - How effective is the recruitment process?
   - Are human resources utilized productively?
   - What are the current and future staffing needs?
   - How can the best people be attracted, retained and motivated?
   - How can individual goals be aligned with corporate strategy?
   - Are the compensation and benefits correctly related to the title and job description?
   - How does compensation relate to market standards?
   - How is attrition managed?
   - What sources do employees have to plan their retirements?
   - Are performance appraisals used efficiently?

3. Key Attributes
   - Employee, Clearance Status, Work Experience, Demographics, Organization, Salary Level, Position Status, Benefit Type, Satisfaction Code, Recruiting Source, Job Start, End Date, Performance Score

4. Key Performance Indicators and Business Metrics
   - Headcount, Salary, Benefit, Attrition, Acquisition, Recruiting, Qualifications, Performance, Average Salary, Vacation Days, Sick Days, Bonus Potential, Bonus Received

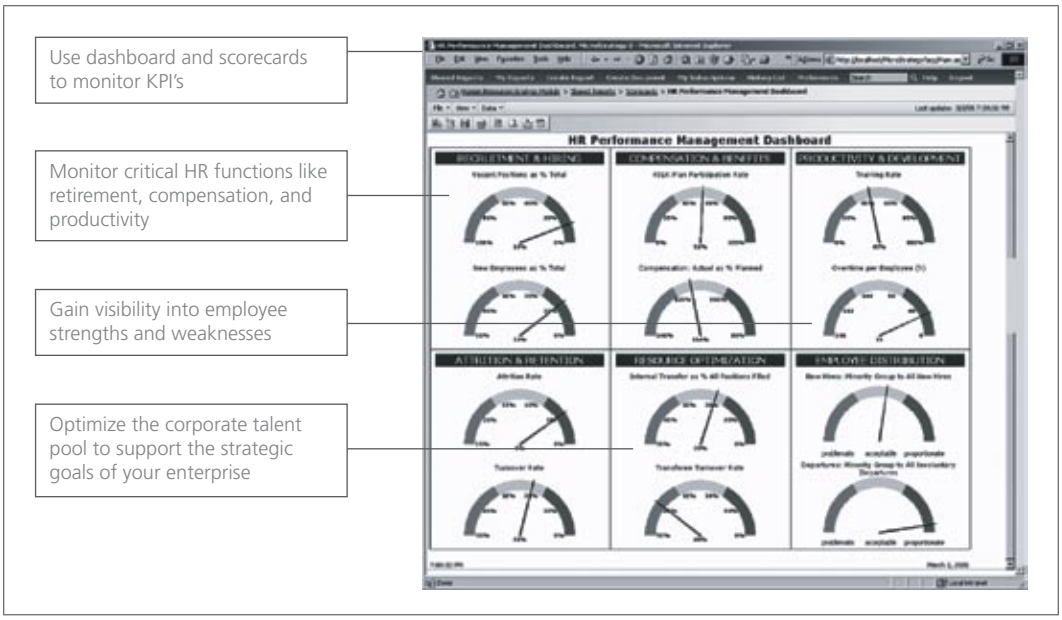


Figure 18.5 MicroStrategy's powerful graphing and visualization capabilities provide HR specialists with the analysis they desire in an intuitive and easy-to-understand format.

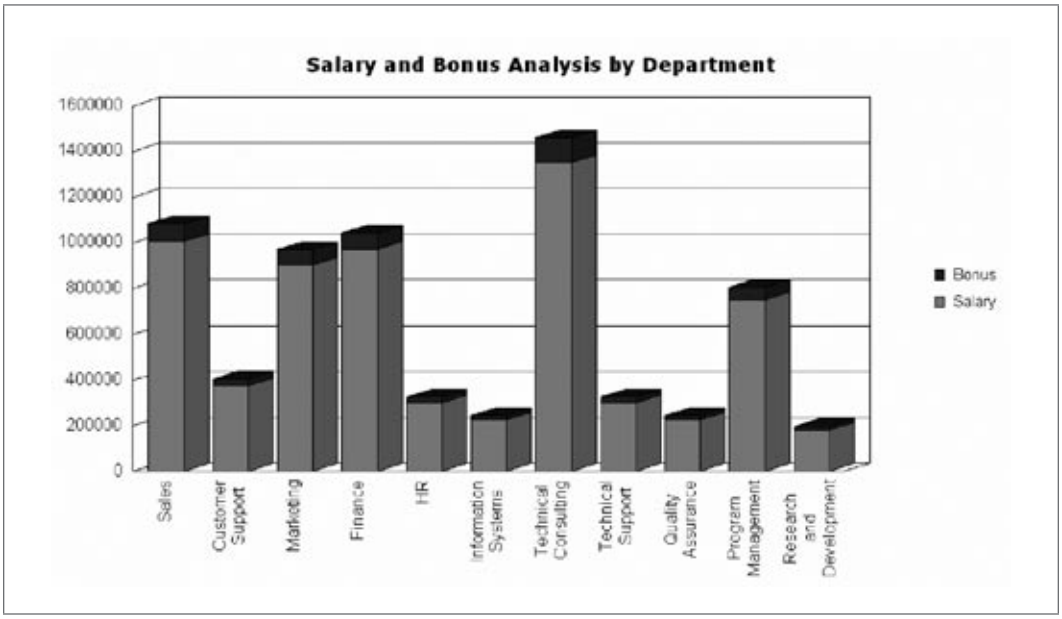


Figure 18.6 Human Resources Analytics Chart.

Sales and Distribution Analysis

The Sales and Distribution Analysis Module includes over 130 reports and 190 key performance indicators and business metrics to help organizations achieve operational efficiencies across the sales-through-distribution value chain. The Sales & Distribution Analysis Module's best practice analytics are fully documented in a 290-plus page reference guide. The analytics and key performance indicators are grouped along five areas of analysis:

1. Key Areas of Analysis
    - Inquiry Analysis
    - Quotation Analysis

- Sales Order Analysis
- Delivery Analysis
- Post-Sales and Returns Analysis

2. Key Questions Answered
   - How are inquires allocated across distribution channels and the sales organization?
   - Which products contribute most to lead generation and inquiry?
   - What is the best strategy to increase the number of inquiries for certain products?
   - Which regions contribute the highest number of quotations, and what is the trend over time?
   - What is the lost opportunity measured as rejected quotations?
   - Which region and product have the highest inquiry-to-quotation conversion?
   - For each quarter, which are my top 5 sales orders by net sales amount?
   - Identify the best and worst performing sales regions and products in terms of profitability, number of sales orders and net margin.
   - Which region and products have the highest quotation-to-sales-order conversion?
   - What is the average delivery time for various products by geography?
   - What percent of deliveries were late? How did they compare with average delays?
   - Which distribution centers process the most orders? Are they more efficient than others?
   - Identify the regions with the best and worst processing times.

3. Key Attributes
   - Distribution Channel, Sales Group, Product, Customer, Customer Geography, Customer Industry, Sales Document, Reference Document, Delivery Document, Shipping Point, Time, Delivery Date, Planned Date, Requested Date

4. Key Performance Indicators and Business Metrics
   - Inquiries, Inquiry Variance, Inquiry Distribution, Average Quantity, Quotations, Quotation Variance, Quotation Distribution, Net Amount, Sales Order, Sales Order Conversion Rate, Sales Order Distribution, Average, Net Price, Discount Price, Cost, Average Delivery Delay

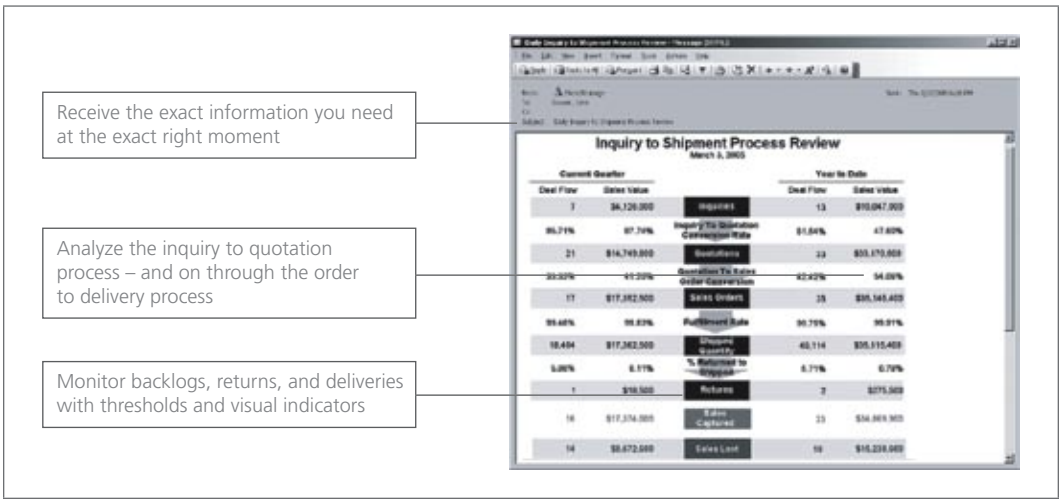


Figure 18.7 Process flow dashboards with MicroStrategy give at-a-glance convenience to reviewing mission critical information. Business users can receive this information directly in their email inbox.

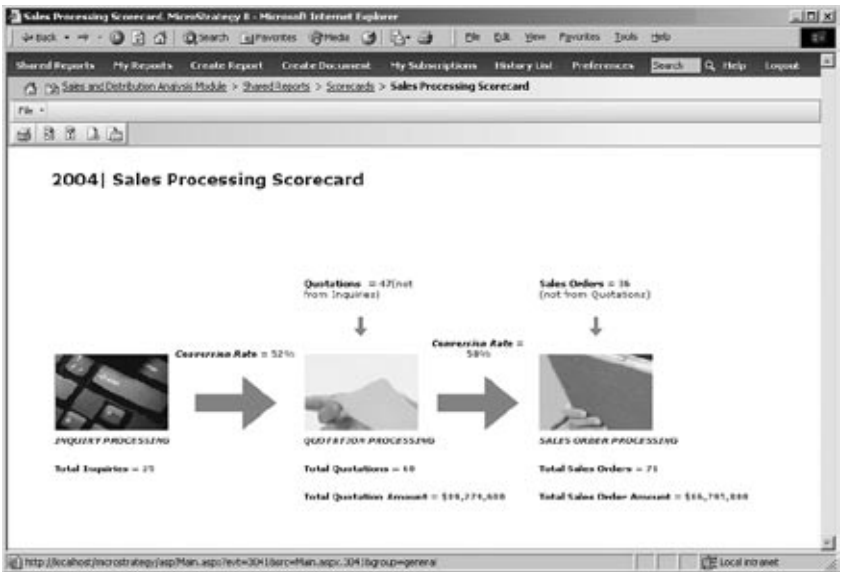


Figure 18.8 Sample Sales and Distribution Analytics Chart.

Sales Force Analysis

The Sales Force Analysis Module includes over 40 reports and 85 key performance indicators and business metrics to help organizations convert leads into sales, monitor sales pipelines, and rank sales executives. The Sales Force Analysis Module's best practice analytics are fully documented in a 200-plus page reference guide.

1. Key Areas of Analysis
   - Lead Type and Quality Analysis
   - Sales Pipeline Analysis
   - Product Sales Analysis
   - Sales Performance Analysis

2. Key Questions Answered
   - What are the sources of my leads and what are the conversion rates for these leads?
   - Are there any trends in leads and lead size?
   - How long does it take to qualify a lead?
   - What is the profile of prospects and customers?
   - What are my expected revenues at any point in time and how does it compare to forecasts?
   - What is the competitor activity in the various accounts and is there a trend?
   - What is the most important reason for lost deals?
   - What are my top selling products and who is buying them?
   - What is the contribution to sales of my top products?
   - Which products have increasing and declining sales momentum?
   - Which products are sold together most often?
   - How are sales representatives performing? Are they on track to meet their sales targets?
   - Which sales regions performed the best and worst? Are there clear trends?
   - Who are the best performers in terms of percentage of opportunities closed and overall revenue?

3. Key Attributes
   - Prospect, Company, Lead Source, Lead, Lead Type, Sales Organization, Opportunity, Order, Time, Competitor, Product, Order, Account or Customer

4. Key Performance Indicators and Business Metrics
   - Lead Conversion Rate, Wins, Lost Deals, Win Percentage, Quota, % Quota Achieved, Opportunities, Pipeline Revenue, Orders, Opportunities, Pipeline Revenue, Target, % of Target.

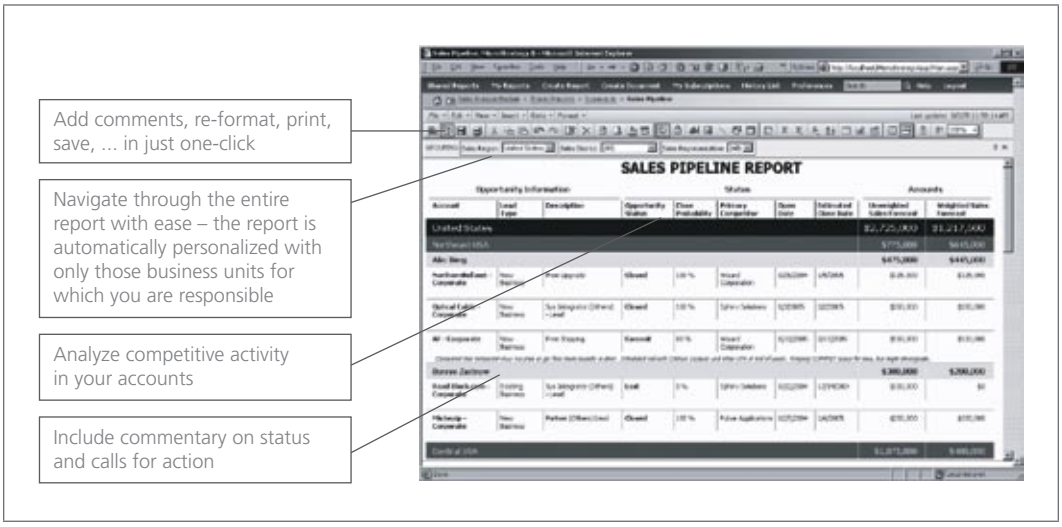


Figure 18.9 Sales performance reports and scorecards in MicroStrategy enable executives to identify top performers as well as areas of concern.

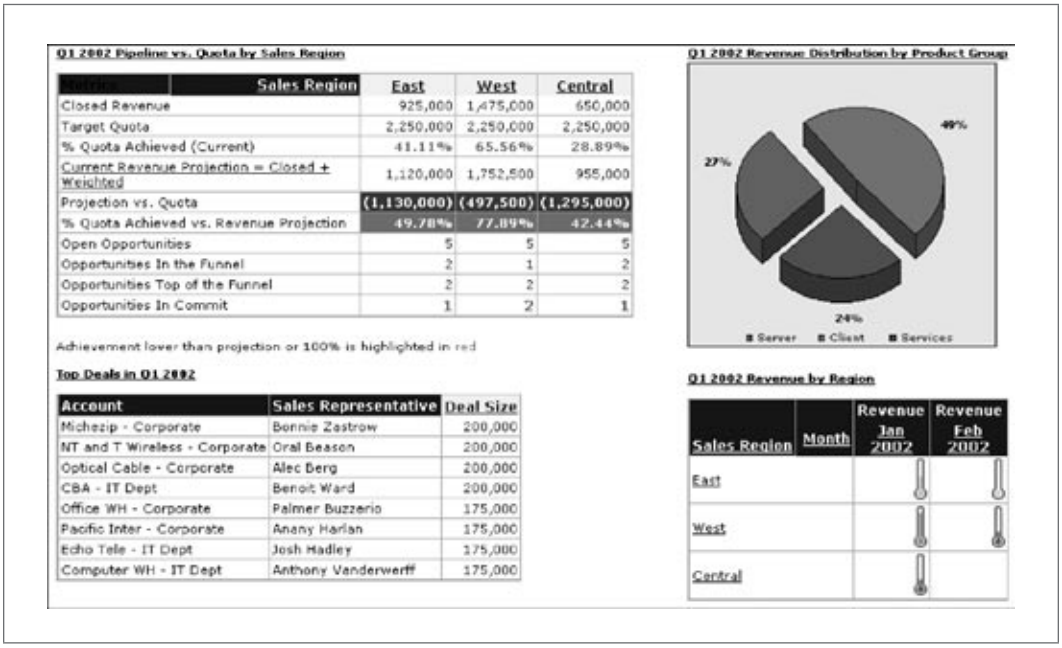


Figure 18.10 Sample Sales Force Analytics Scorecard.

Web Traffic Analysis

The Web Traffic Analysis Module includes over 60 reports and 80 key performance indicators and business metrics that provide deep insight into the branding, ad campaigns, and target markets based on detailed data captured by the corporate website. The Web Traffic Analysis Module's best practice analytics are fully documented in a 190-plus page reference guide.

1. Key Areas of Analysis
   - Web Traffic Highlights
   - Web Content Analysis
   - Web Visitor Analysis

2. Key Questions Answered
   - What is the trend across Web sites for different time periods?
   - Are there any unusual, non-seasonal changes in trends that need further investigation?
   - Which business scenarios are providing me with positive outcomes?
   - Which banner ads and referrals are attracting qualified visitors?
   - What is the success rate in converting visitors to loyal customers?
   - How loyal are my customers? Which customers are defecting?
   - How are first-time visitors using the site vs. repeat visitors?
   - How can you optimize your page sequences for best conversion rates?
   - What pages cause visitors to drop out?

3. Key Attributes
   - Web Sites, Visitors, Visit Types, Events, Referrers, Banner Ads, Internet Search Keywords, Page URL, Page Title, Content, Paths, Local Search Keywords

4. Key Performance Indicators and Business Metrics
   - Page Views, Visits, Traffic Counts, Stickiness, Click-Through Counts, Conversions, Advanced Statistics, Traffic Counts

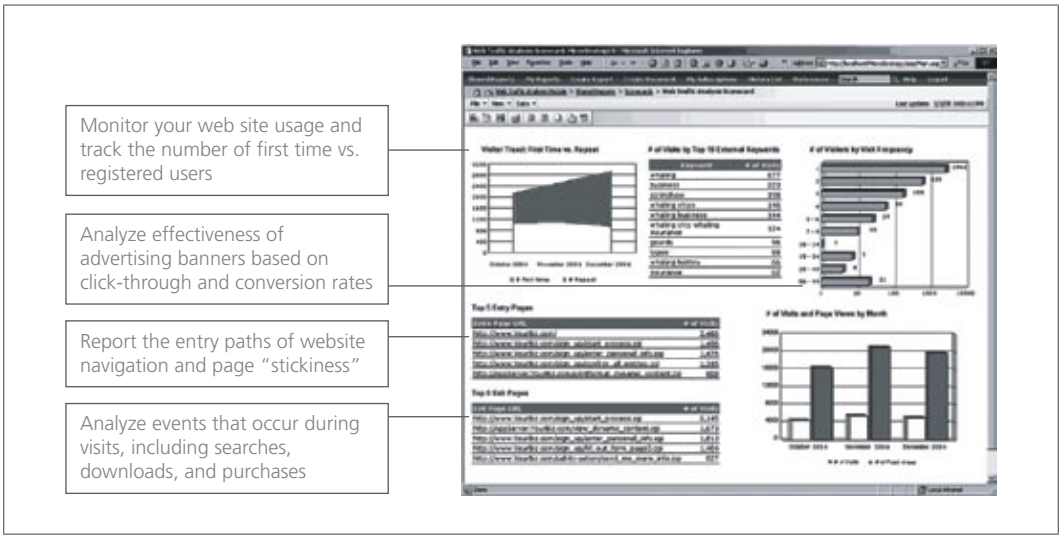


Figure 18.11 No other interaction can yield as much detailed information about your audience as Web visitor behavior. With MicroStrategy, each piece of knowledge is easily understood, analyzed, and monitored.

18.4 Summary

The MicroStrategy BI Developer Kit assists in rapid development of corporate performance management applications, delivering analytic starter modules and a complete MicroStrategy development environment.

Each application module is designed as a starter kit complete with scorecards, dashboards, reports, metrics, and KPIs. Business users access these analytics easily from: any Web browser, email program, or Microsoft Excel, PowerPoint, Word, and Outlook.

Each module includes an application model (logical schema), a data warehouse model (physical schema), and extensive documentation. MicroStrategy Desktop Designer enables full extensibility and customizability. MicroStrategy Architect maps the logical schema to any physical schema. MicroStrategy BI Developer Kit customers take full advantage of their existing data warehouse and ERP investments, and are able to develop applications rapidly.

MicroStrategy's application modules are built on the MicroStrategy platform, taking full advantage of all 5 styles of business intelligence, data scalability, user scalability, and centralized IT infrastructure. The MicroStrategy BI Developer Kit is unique because it combines the flexibility of building custom applications with the speed of deploying packaged applications, at a price well below the six-figure BI applications available in the market from other software companies.

19

Extensibility and the SDK

A key component of an enterprise-class software platform is the ability to modify and extend existing functionality to meet the unique needs of an organization. A robust BI platform must provide the tools necessary to leverage built-in functionality, and must allow access to all functionality through application programming interfaces (APIs) and a variety of programming languages. A thoroughly documented API should enable easy customization and extension of application functionality. Finally, a platform that serves as the basis for application development should include sample code and utilities that give guidance for modifying and extending the platform.

Three components in the MicroStrategy BI platform make customizations and extensions to any BI application possible and easy to maintain:

- A Service-Oriented Architecture and open API
- Documentation enabling access to the API and platform functionality
- Development kits, utilities and sample code

19.1 Access All BI Functionality in a Service-Oriented Architecture with Open APIs

The MicroStrategy BI platform is a modern, sophisticated service-oriented architecture designed for flexibility and extensibility so that it can grow with the varied demands of enterprise applications. Due to the openness of the MicroStrategy BI platform, external applications can instantly take advantage of BI functionality, performance, failover, load balancing, and end-user interfaces by integrating or plugging directly into the platform. The ability to plug into the MicroStrategy BI platform enables extensive user experience customizations and product innovations that help organizations make the most effective use of information to drive business performance.

MicroStrategy's organic service-oriented architecture guarantees efficiency at each service or component with no code redundancy. This organic architecture is in stark contrast to the service-oriented architectures claimed by other BI vendors that were necessitated when mergers led to disparate, separately developed architectures being pieced together. Unlike SOA's of other BI vendors that wrap code to hide redundant functionality and inconsistency across components from acquired technologies, the organic MicroStrategy architecture exposes consistent, encapsulated BI functionality using thousands of documented interfaces and methods.

An API forms the mechanism to access data and functionality from outside the MicroStrategy platform. The platform includes the following groups of API's:

- MicroStrategy Web API
- Intelligence Server API

- Narrowcast Server API
- MicroStrategy Office API
- Transactor API

These API's are used to customize and extend the functionality of the MicroStrategy platform, build new applications, or embed MicroStrategy functionality into other applications.

19.2 MicroStrategy Web API

Most organizations that customize the MicroStrategy BI platform use the Web API. The Web API allows customization of the MicroStrategy Web interface and development of custom Web applications that provide interactive reporting functionality to any browser, portal or third-party Web application.

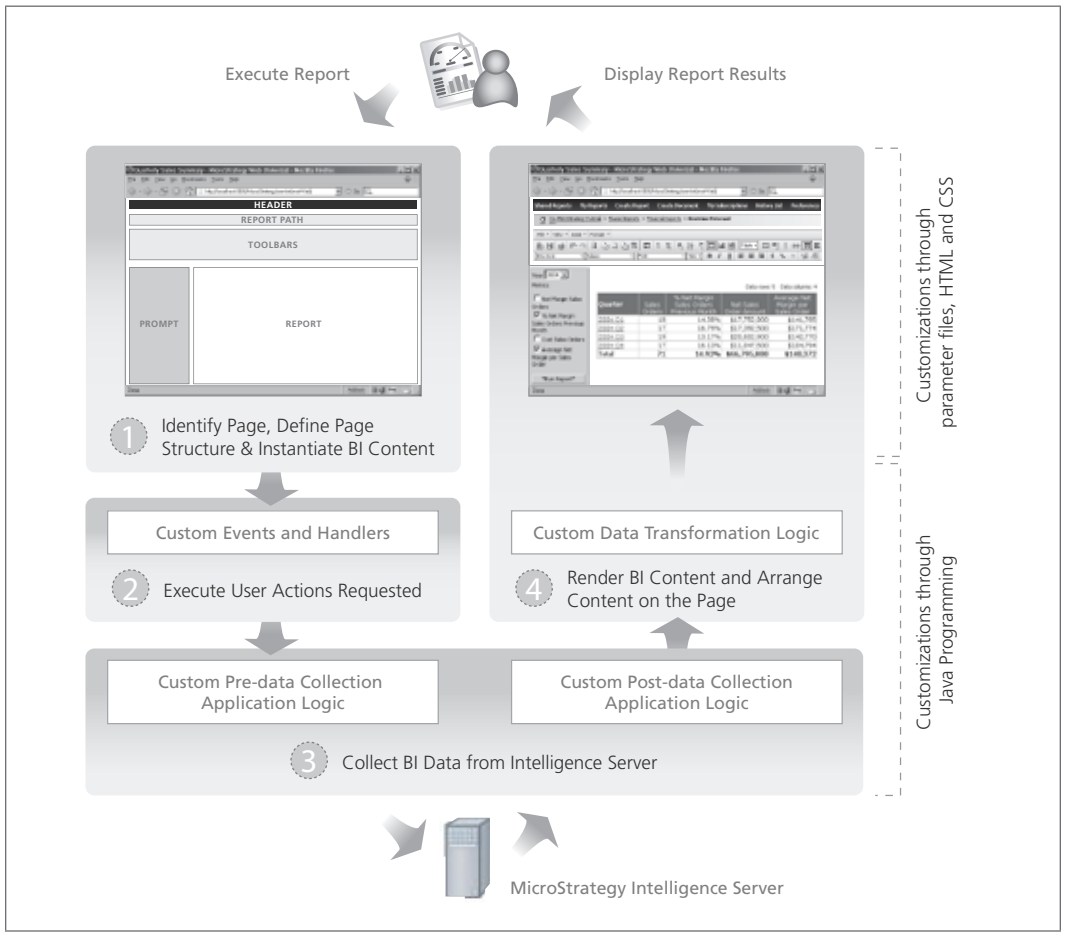


Figure 19.1 Web API enables extensive customization of every stage of the page execution workflow.

The Web application workflow involves four distinct stages:

1. Define the page structure, and instantiate BI contents displayed on the page
2. Execute user actions requested
3. Collect BI data from Intelligence Server
4. Render BI content, and arrange content on the page

The first and last steps of the Web application workflow define layout of the Web interface, and present visually appealing rendering of BI content such as folder contents, report data, prompts, schedules, and subscriptions. Easy to edit and maintain XML configuration files, Cascading Style Sheets (CSS), and xHTML Layout Definition files specify the page definition, and rendering of reports and other BI content on any page. No ASP.NET, JSP, or Java coding is required for most presentation-related customizations.

The second and third steps of the page execution workflow define the application functionality. An example of functionality customization is the integration of a mapping tool to display geospatial visualization of business data. Such advanced functionality customizations require Java programming skills.

Modify Page Layout and Formatting Using Parameter Files

The layered Web architecture and comprehensive Web API allow developers to customize the structure, contents, layout, and formatting of every Web page through externalized parameter files. The benefits of externalization of common customization settings into configurable Parameter (XML), Style (Cascading Style Sheets), and Layout Definition (xHTML) files include:

- Easy customization of the application through textual changes to the parameters in configuration files
- Reduced programming skill requirement for the most common customizations. Modifying configuration files, CSS, and Layout files does not require Java or .NET programming
- Seamless upgrades to future versions of MicroStrategy Web. The configuration files provide a ready catalog of all changes made to the default configuration of MicroStrategy Web, resulting in customizations that are easy to maintain, extend, or roll back.

MicroStrategy Web provides numerous preconfigured pages specifically tailored to the activity performed on each page. The default pages in MicroStrategy Web include a Login/Logout page, a Shared reports a page, a My Reports page, a Report Execution page, a Create Report page, a Create Document page, and a My Subscriptions page, among others.

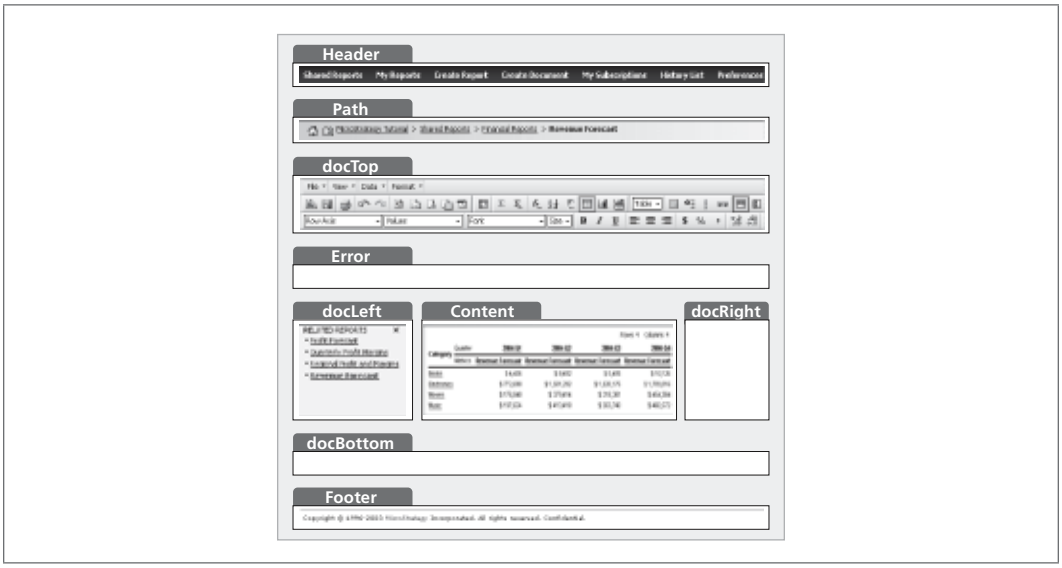


Figure 19.2 A schematic representation of the Report Execution page displays the various page sections that comprise the page.

Every page of MicroStrategy Web has a well-defined structure containing numerous page sections. These page sections and the content displayed within them are defined independent of each other, giving developers the utmost design flexibility and application functionality. Additionally, each page can have multiple states, including an initial state, a final state, and any intermediate states that show the progress of the requested actions.

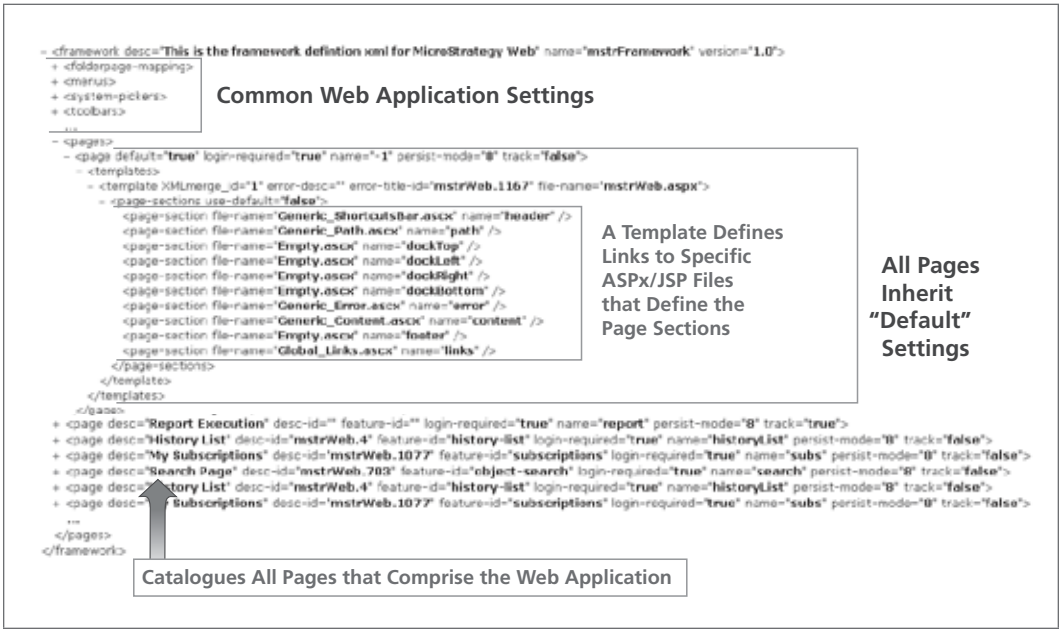


Figure 19.3 A Page configuration XML file defines the framework of the MicroStrategy Web application. It lists all the pages, the associated page settings, and common application settings such as mapping of system folders to pages, menus, and toolbars.

The Page Configuration file catalogs all pages available to the Web application in an easy to understand and manageable format. In addition to customizing the page structure and BI content of default Web pages, organizations can add custom Web pages very easily into the application by referencing the new pages in the Page Configuration file.

MicroStrategy Web provides an extensive set of Transforms that render BI information retrieved from Intelligence Server into viewable formats (HTML) for display in the Web interface. The Transforms use Styles and Layout Definition files to specify the presentation of BI content and its arrangement on a Web page. A Layout Definition file embeds MicroStrategy-specific tags into a familiar HTML structure. Externalization of presentation-related customizations into layout definition files and cascading style sheets (CSS) enables easy maintenance and upgradeability of customizations.

A centralized XML configuration file called Style Catalog (styleCatalog.xml) references all transforms, styles, and layout definitions available to an application.

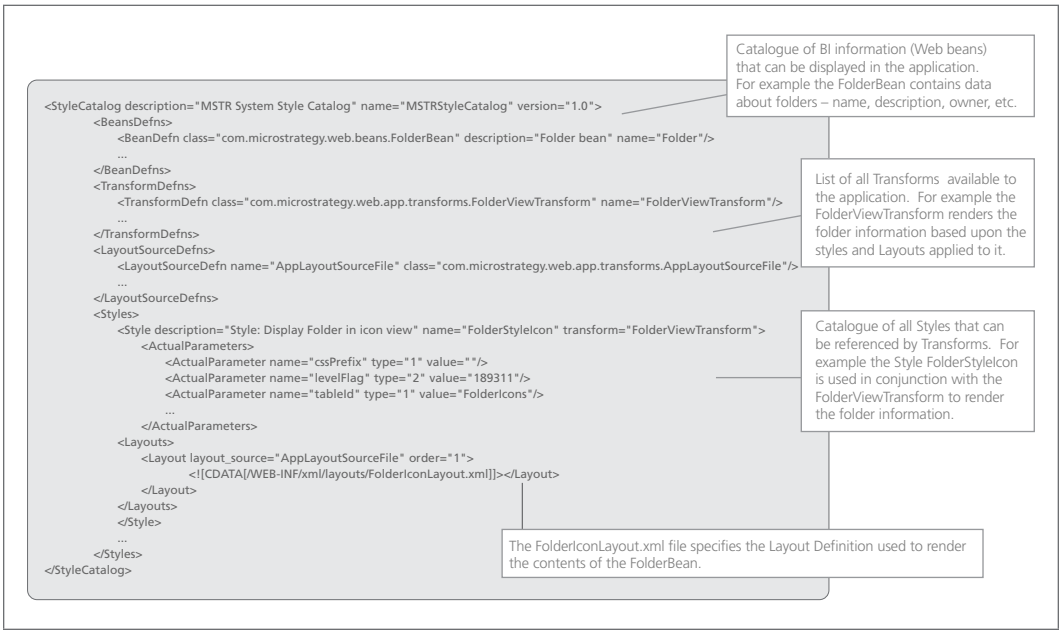


Figure 19.4 All the Transforms, Styles, and Layout files are cataloged in an easy to change XML configuration file called the Style Catalog.

Extend Application Functionality through Java Programming

Three categories of customizations using the Web API enable highly customized application logic and presentation formats:

- Functionality customizations that affect requests only on a specific page
- Functionality customizations that affect requests on all pages of the application
- Advanced transformation of BI content to alternate presentation formats

Each of these three categories of customization requires Java programming.

User actions that request specific application functionality, such as clicking on a report to execute a report, or clicking on a change-to-graph view icon represent "events" to the Web API. Events trigger the set of application tasks that modify the state of individual BI components or Web beans (Java beans) on a page. They occur before information requests to the Intelligence Server. An event handler manages the execution of the event. For example, event handlers specify how to process user actions such as execute reports, change report formatting, sort report results, and pivot rows and columns of a grid. Custom application functionality that is specific to particular pages can be implemented through existing MicroStrategy Web event handlers or through new ones.

Unlike events, "add-ons" deliver application functionality that requires specific behavior every time any MicroStrategy Web page is loaded. Examples of common tasks performed by add-ons in MicroStrategy Web include setting execution and result flags, and specifying rules for incremental fetch on grids and graphs. Add-ons execute custom code before or after data are retrieved from Intelligence Server.

The third category of Web customizations that require Java coding involve extending existing transforms or creating new transforms. Custom transforms display BI information retrieved from Intelligence Server in custom presentation formats required for the BI application. For example, custom transforms can render report results in Wireless Markup Language (WML), plain text, or binary data instead of HTML.

19.3 MicroStrategy Intelligence Server API

MicroStrategy Intelligence Server contains an API that can access many aspects of application modeling and development, report execution and data manipulation, and platform administration and monitoring. The Intelligence Server API has the following features:

- Fully thread-safe
- Object-oriented, with more than 3,000 documented methods and properties
- Any third-party Windows application can access Intelligence Server running on the Windows, Linux, or UNIX platforms.

The main customizations performed using the Intelligence Server API involve integration with system management software.

19.4 MicroStrategy Narrowcast Server API

The Narrowcast Server API exposes the information delivery and subscription functionality of MicroStrategy Narrowcast Server.

It consists of three groups of API's:

- Subscription API
- Delivery Engine API
- Narrowcast Server

The Subscription API facilitates the development of custom Web applications that enable subscription to report delivery services from a corporate Web page, rather than through Narrowcast Server's out-of-the-box Subscription Portal. It encapsulates various operations involved with creating, editing, or deleting subscriptions or their associated components.

The Delivery Engine API handles all tasks related to the Delivery Engine component of Narrowcast Server. These include:

- Start and stop the Narrowcast Server Delivery Engine
- Connect/disconnect from the object repositories and the primary distribution manager to obtain system information
- Query the status of the various components of Narrowcast Server
- Authorize user actions
- Execute report delivery services on demand
- Pause or resume service execution

The Narrowcast Server API extends the capability of the Narrowcast Server to suit custom application requirements. The typical extensions include creating new:

- Information Source Modules that retrieve data from sources other than Intelligence Server, such as unstructured data and ERP data
- Document Formatting Modules to format content included in the delivery services
- Information Transmission Module that transmit content to new output devices, such as PDA's or wireless devices

19.5 MicroStrategy Office API

The MicroStrategy Office API allows developers to build custom applications that use Microsoft Office products as a user interface to the MicroStrategy BI platform. The MicroStrategy Office API is used to create powerful Word, Excel, and PowerPoint applications that incorporate insight gleaned by the MicroStrategy BI platform. The API facilitates access to multiple MicroStrategy projects, selection of one or more reports, automatic prompts answering, and embedding of report results into Microsoft Office applications.

19.6 MicroStrategy Transactor API

MicroStrategy Transactor provides fully documented Java APIs to allow developers to embed Transactor's write-back and closed-loop transaction functionality into applications such as a Web-based reporting and analysis applications. For example, developers can use Transactor API to build custom drivers that update information in ERP and legacy systems directly from business intelligence Web reports or emails.

19.7 Comprehensive Documentation that Describes the API in Detail

A key component of an open platform is comprehensive documentation that describes the methods and properties used to access the objects and functionality of the platform. The MicroStrategy Developer Library provides:

- A comprehensive documentation of the various aspects of the MicroStrategy BI platform
- API references for the various categories of API discussed in the previous section
- Over 80 task-oriented customization and integration examples with sample code

All of these are available in a searchable and indexed HTML format that is easy to navigate.

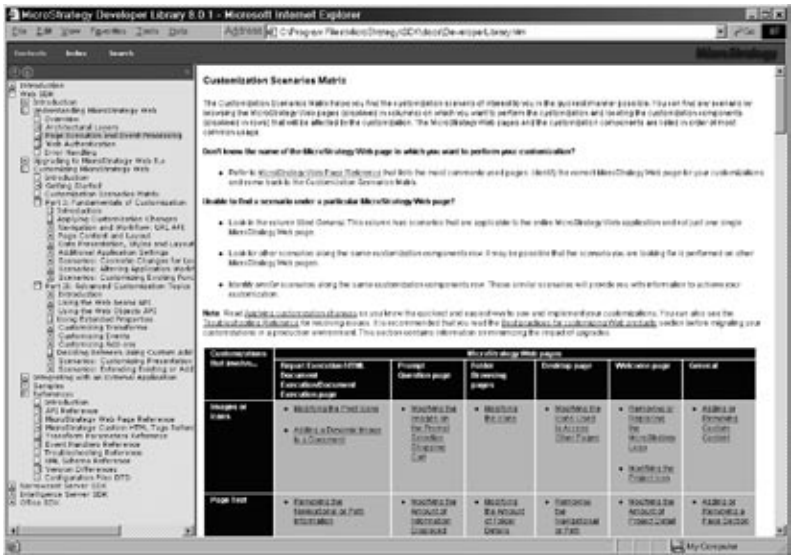


Figure 19.5 The MicroStrategy Developer Library describes customization and integration techniques.

Topics covered in the MicroStrategy Developer Library include:

- Connecting to a BI application and browsing for reports, filters, and metrics
- Modifying the navigation and workflow of the Web interface
- Modifying the page layouts and content displayed
- Running reports and documents, and retrieving report results
- Retrieving prompt parameters, and automatically answering prompts to specify report parameters
- Manipulating report results (drilling, sorting, pivoting, etc.)
- Implementing single sign-on (SSO)
- Setting up customized authentication and validation security paradigms
- Performing user management tasks such as creating users, organizing them in user groups, creating security filters, and assigning them to users
- Integrating with Portal products such as SAP Netweaver Portal®, IBM® Websphere®, Microsoft Sharepoint®, and other JSR-168 compliant portals
- Integration with other external applications

The API Reference provides detailed information of all packages along with a summary of classes and interfaces. The reference also lists the class hierarchies for the packages that comprise the API.

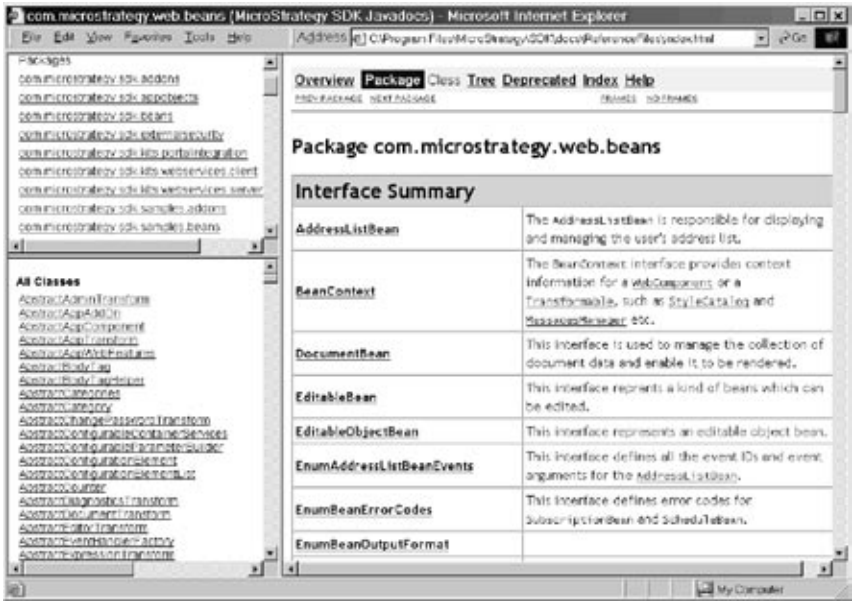


Figure 19.6 The MSDL provides a comprehensive reference of all categories of API using interactive HTML-based Java Docs.

19.8 Simplify Development through Utilities and Sample Code

Any open and extensible platform should include development utilities that aid developers in creating and deploying applications quickly and efficiently. The MicroStrategy SDK contains several such applications and tools. These include:

- Web Customization Upgrade Utility
- CSS Analyzer
- Portal Integration Kit

- Web Services Development Kit
- Sample applications
- Sample code

Web Customization Upgrade Utility

The Web Customization Upgrade Utility compares the customizations made to the Page Configuration and Style Catalog files of an older MicroStrategy platform version to those available with new versions.

The original version's configuration file serves as the baseline for identifying changes made in the customized configuration file. A similar comparison highlights changes between the baseline and the configuration file of the new release. A result pane lists all changes in a color-coded format for easy identification.

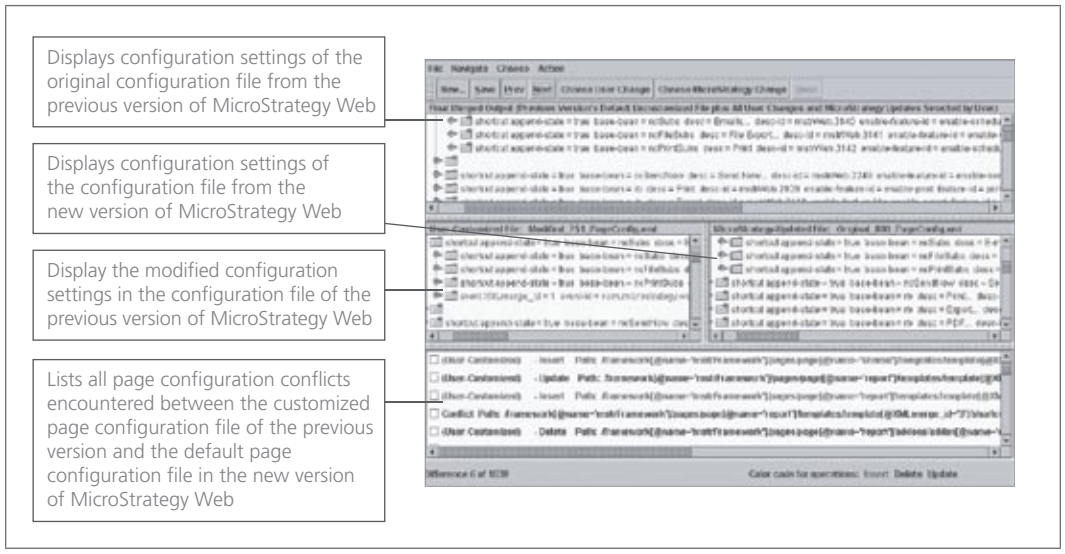


Figure 19.7 The Web Customization Upgrade Utility upgrades customizations of the page configuration to new releases of the Web interface.

Selecting items in the comparison results screen opens up the respective configuration files to the exact location where the upgrade utility encountered differences. The easy identification of customized parameters and automatic cursor placement in the appropriate configuration files makes upgrading Web customizations very easy and manageable.

CSS Analyzer

Cascading Style Sheet (CSS) files define many of the look and feel elements of MicroStrategy Web, such as fonts, font sizes, colors, images, and layout styles for prompts. MicroStrategy Web uses CSS files in three categories:

- Application-wide formatting
- Page-specific formatting
- Browser specific formatting

While using CSS to define formatting rules reduces the size of HTML downloaded to a browser client, the cost is in keeping track of the CSS file that introduced the formatting. The SDK includes a CSS Analyzer utility that identifies the specific CSS definition used by a given HTML element on any MicroStrategy Web page through point and click actions on the Web interface. Making changes to CSS files is very straightforward once the specific CSS file and selector are identified.

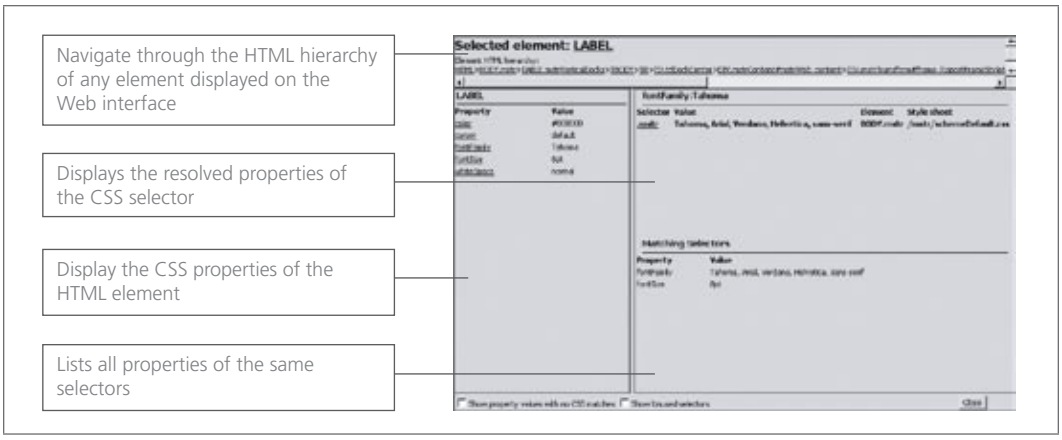


Figure 19.8 The CSS Analyzer identifies cascading style sheet (CSS) definitions used by a given element on any MicroStrategy Web page through point and click actions.

19.9 Portal Integration

The MicroStrategy SDK provides detailed instructions necessary to incorporate the full range of BI functionality, formatting, and user interactivity available to the MicroStrategy Web interface in any commercial or homegrown portal. The integration with portal technologies stems from MicroStrategy conformance to industry standards, such as J2EE, .NET, XML, JSR 168, WSRP, and SOAP. The SDK includes the following portal integration capabilities.

Deployment-ready MicroStrategy Portlets

Deployment-ready portlets require no additional programming when delivering MicroStrategy Web reports and functionality in the following third-party portal products

- IBM WebSphere Portal
- SAP Netweaver Portal
- Microsoft SharePoint Portal

The MicroStrategy portlets embed folders, reports, documents, user history lists, and a search page into the portals through easy-to-configure screens. The portlets provide the full range of OLAP manipulations such as sort, pivot, add subtotals, export, and add new calculations, as well as design functionalities such as changing the report display between grids and graphs, and toggling thresholds.

The SDK contains detailed instructions for the installation, configuration, and deployment of the portlets. The portlets leverage the portal's credential storage mechanism to perform single sign-on. In addition, they also provide personalization by saving and retrieving user-specific information from the portal repository, and making it available to MicroStrategy at runtime.

Sample Code for Integration with Other Portals

A Portal Integration Kit includes sample code and documentation for integrating MicroStrategy Web with other enterprise portals. It illustrates the integration architecture, workflow, and usage of the API in the context of deploying Web functionality to portals. In addition, the SDK includes sample code to embed MicroStrategy Web functionality into any portlet that conforms to the Java Specification Request (JSR) 168 Portlet Specification.

Sample code for deploying MicroStrategy within the Apache Struts framework Struts (http://struts.apache.org/) provide an open source framework for building Java Web applications. The core of the Struts framework is a flexible

control layer based on standard technologies such as Java Servlets, JavaBeans, ResourceBundles, and XML. The SDK provides sample code to incorporate MicroStrategy functionality within a struts framework.

19.10 Web Services Development

Web Services allow loosely coupled integration with other applications. External applications can use Simple Object Access Protocol (SOAP) to access any MicroStrategy functionality. A Web Services Development Kit composed of sample code and documentation describes MicroStrategy's Web services API. It provides a starting point for the integration of MicroStrategy reports with other Web applications using Web Services.

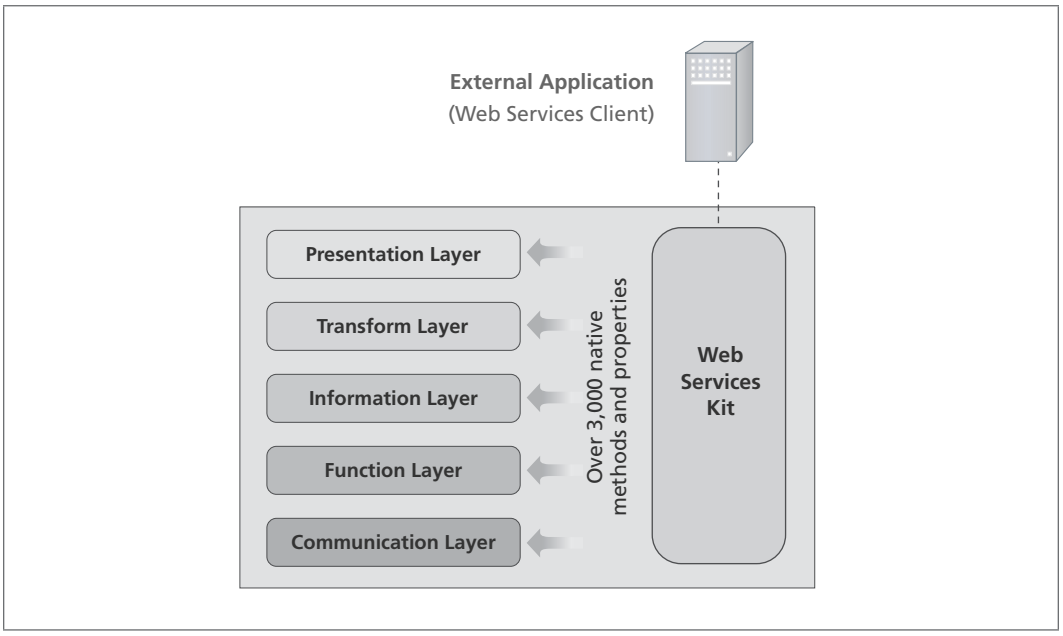


Figure 19.9 The Web Services Development Kit can be used to integrate BI applications with external applications using Web Services.

Some common functionality available through Web Services includes:

- Connect to and disconnect from the MicroStrategy-based application
- Browse for objects
- Answer prompts
- Retrieve attribute element lists
- Run reports and documents
- Incrementally fetch more results
- Drill for more data
- Retrieve the history list for a user

19.11 Sample Applications

The MicroStrategy SDK includes several sample applications that assist in the development process or administration of the BI infrastructure. These include:

- User Manager
  An application that manages user accounts through the Web. Using this application, the administrator can

create new users or groups, or modify privileges, assign security filters, and manage other authentication information for existing users and groups.

- External Security Module
  An application that demonstrates integration with external security sources for customized authentication purposes.

- Session Helper Bridge
  The Session Helper Bridge sample application illustrates how to create a session that connects to MicroStrategy Intelligence Server in a .NET environment. Such customizations may be required to embed MicroStrategy functionality within a .NET-based portal Server.

- Extended Property Editor
  Any object in the MicroStrategy metadata can store additional information in Extended Properties. For example, a developer can create an extended property on all reports that enables end users to write report data back to the data warehouse. The Extended Property Editor allows any developer to associate a series of extended properties with any metadata object.

Sample Code

MicroStrategy SDK includes the source code for the sample applications and utilities. This sample code is available in the following languages:

- Java
- .NET
- Visual Basic®
- Visual C++®

The SDK includes additional code samples in the MicroStrategy Developer Library.



Figure 19.10 The MicroStrategy Developer Library contains over 80 task-oriented examples with sample code.

Categories of sample code in the MicroStrategy Developer Library include:

- Add-ons: Sample programs that illustrate the use of add-ons to answer prompts based upon URL query string contents, and redirects to remote URL's
- Events: Sample code that illustrates how to add custom logic upon modification of a user's password, event handlers for reports, prompts, and Web bean-specific events
- Transforms: Sample code to edit, modify, and display contents in context-sensitive menus
- Security: Sample code to implement single sign-on solutions with Identity Management applications and user validation

19.12 Customization and Extension Examples

Over half of all MicroStrategy customers, customize or integrate their MicroStrategy platform to some extent. The range of customization varies widely. Some customizations entail altering the MicroStrategy Web user interface to adhere to corporate look-and-feel guidelines. Other customizations add new user functionality directly to the MicroStrategy code set. Under a third type, customers embed MicroStrategy functionality deep within other applications that they have written or purchased.

As a result, customizability, and functional and system extensibility have been a major thrust of MicroStrategy engineering for the past several years. Specifically, MicroStrategy is adding more and more features to make MicroStrategy customizations easy to implement and maintain.

Customization Ease	Common Customization Goals	Enabling MicroStrategy Feature
Out-of-the-box	Embed MicroStrategy Web in a Portal	Portal Integration Kit
	Embed MicroStrategy Web in other Web Applications	Zero footprint Frame Integration
Parameter-driven	Modify the look and feel of the Web Interface. Define custom layouts and formats for BI data such as prompts and reports.	Configuration files, CSS, and Layout Definition files
	Provide advanced rendering of BI content and new application functionality	J2EE™ compliant architecture with specialized Java constructs – Transforms, Events & Add-ons
Basic Programming	Implement Single Sign-On (SSO) solutions	External Security Module
	Create integrated Microsoft Office applications	Web Services & MicroStrategy Office API
Sample Code & Tools	Access all MicroStrategy functionality from other applications	Service-Oriented Architecture with 3000+ methods and properties
Full Programmatic Control	Integrate with system control and administration tools	Functionally rich Intelligence Server API

Figure 19.11 Common customizations and extensions performed on the MicroStrategy BI platform.

Embed MicroStrategy Web in Enterprise Portals

Organizations can extend the reach of BI reporting and analysis by deploying the full range of BI functionality, formatting, and user interactivity available to the MicroStrategy Web interface in any commercial or homegrown portal.

Embedding multiple MicroStrategy portlets on a single portal page offers organizations the ability to create dashboard-style applications. Each MicroStrategy portlet optionally calls a customized report execution page with just the interactive reporting capabilities required. MicroStrategy portlets provide the full range of functionality of MicroStrategy Web. These include:

- Displaying toolbars to change views of reports, format, and export data;
- Drilling for more details within the portlet or to a new window;
- Enabling context-sensitive right-click menus;
- Performing OLAP manipulations such sort, pivot, page by slices of the report, and add new calculations.

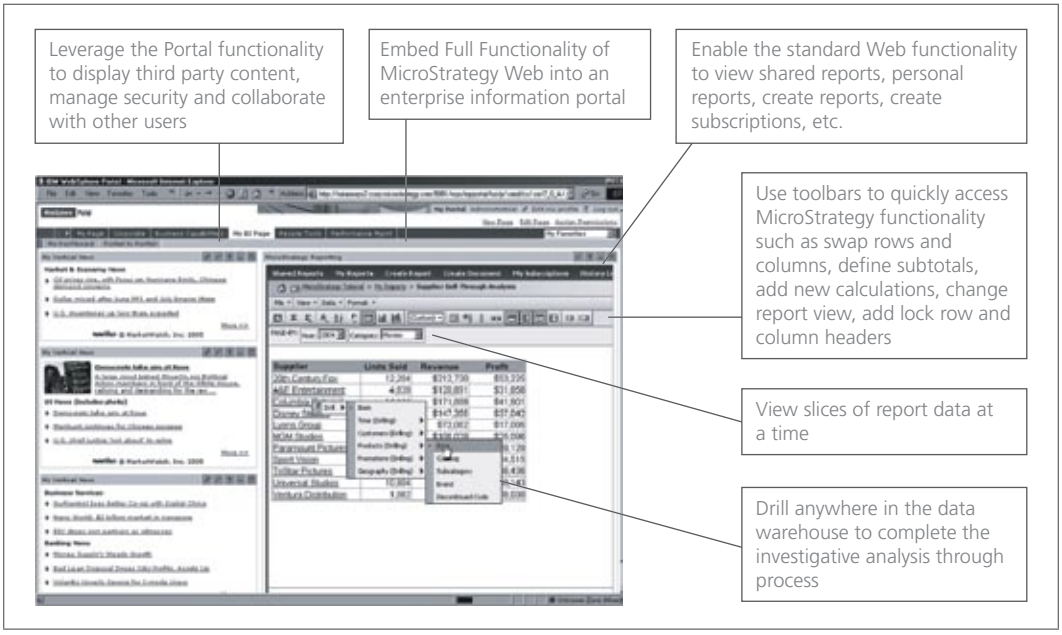


Figure 19.12 Embed the full range of MicroStrategy functionality into enterprise portals using deployment-ready portlets or Web parts, while leveraging the portal's security and collaborative capabilities.

On the other end of the interactivity spectrum, a portlet can provide alert-style reports with no user interactivity.

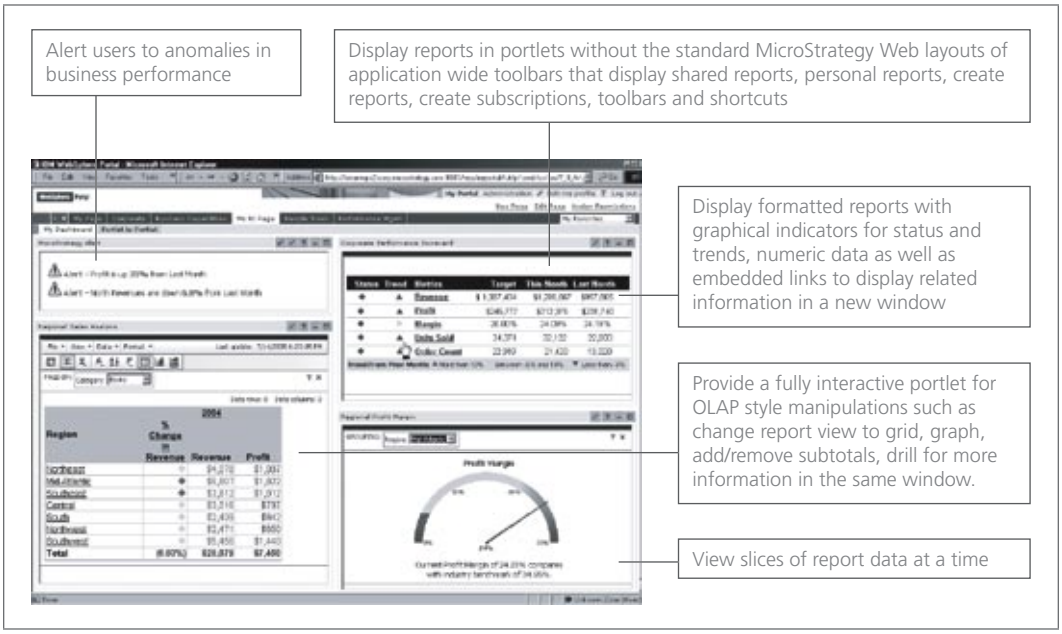


Figure 19.13 Create powerful dashboards that incorporate content from multiple data sources while providing a wide variety of end user functionality within individual portlets that comprise the dashboard.

Another requirement in portal applications is the ability to control the content displayed, and even the application logic of other applications based upon the state of a "master," or controlling portlet. MicroStrategy portlets pass parameters such as report names, prompts, attribute elements, and metrics values to "slave" portlets that use the information for further processing. A simple illustration of this scenario involves a controlling portlet that displays report folders and report lists. Clicking on the report name in the master portlet displays the report results in one of the slave portlets.

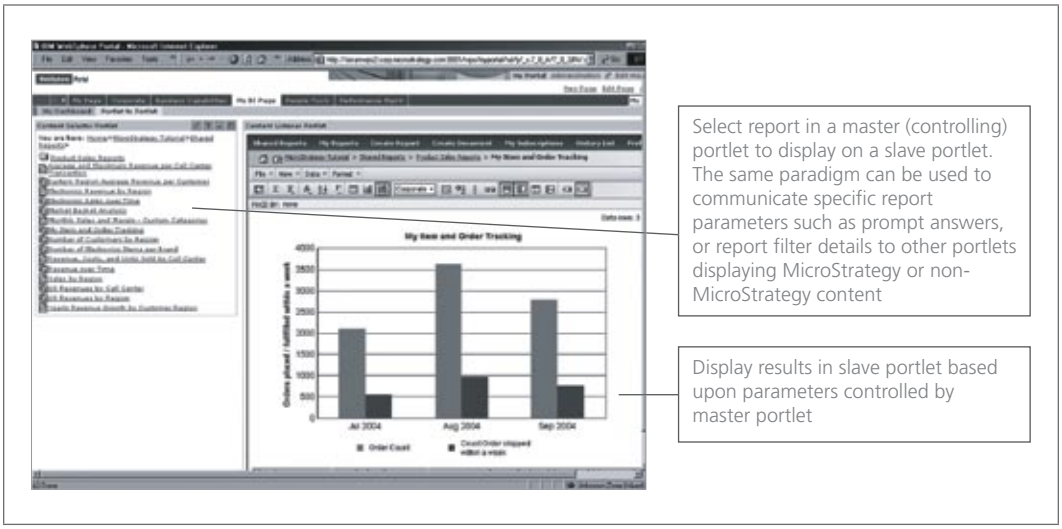


Figure 19.14 Build powerful portal applications that use MicroStrategy content such as report names, prompt answers, filter selections to control the information and actions of one or more slave portlets.

Embed MicroStrategy Web in Other Web Applications

Third-party applications easily incorporate a significant portion of MicroStrategy Web functionality through appropriately constructed URL's. OEM partners of MicroStrategy use their industry expertise to create custom analytic applications using the MicroStrategy platform. They then package these applications and sell them as their own branded products. This simple approach to integrate MicroStrategy with external applications serves numerous use cases:

- Link external applications or websites to any MicroStrategy application in the organization
- Browse contents of folders
- Execute reports and documents

Define custom report workflow using linked drilling from a document to a report or other documents.

- Display complete MicroStrategy pages within third-party applications by passing the appropriate URL parameters from an embedded inline frame.
- Serve BI data to third-party applications in any format such as HTML, XML, and comma/tab-delimited.

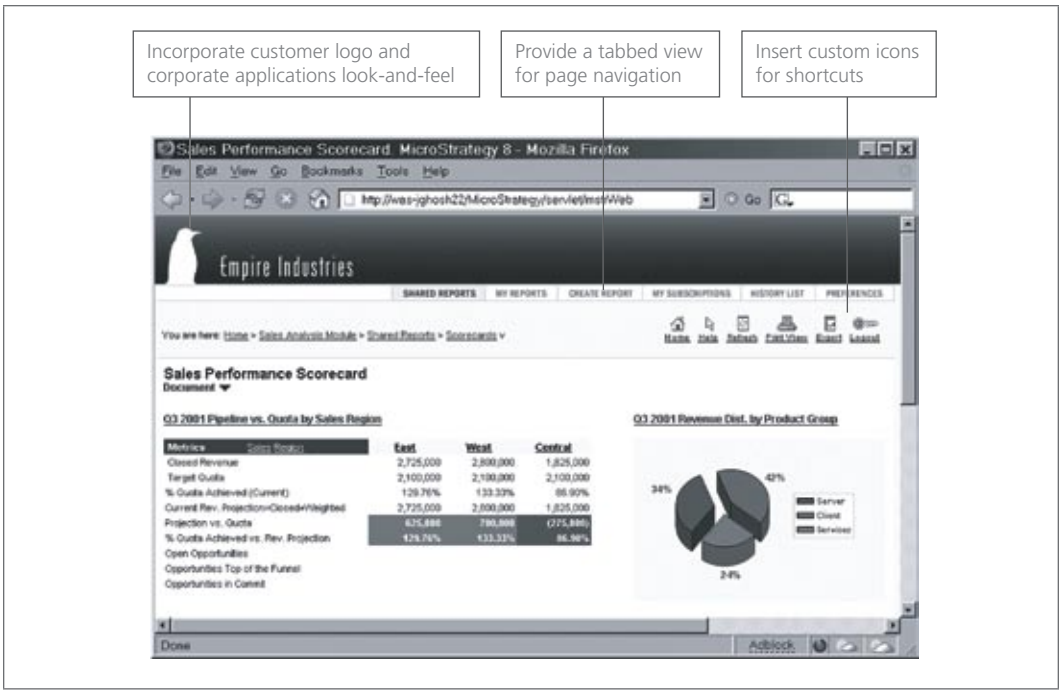


Figure 19.15 Perform look and feel customizations such as including a customer logo, and modifying the page navigation easily using simple modifications of the parameter files and cascading style sheets.

Modify the Look and Feel of the Web Interface

The most common customizations of the Web interface are look-and-feel changes such as matching corporate standards, changing the content of the default MicroStrategy Web page and section layout or adding links to other Web sites. With MicroStrategy, one can accomplish most look-and-feel customizations by changing parameters in the Page Configuration and Style Catalog files. In addition to these configuration files, MicroStrategy uses CSS and Layout definition files that allow application developers to apply formatting styles to the various elements displayed on a Web page. No Java or .NET programming is required to make these changes. As a result, customizations are easier to implement, deployment times are reduced, and support costs are lower.

Advanced functional customizations of the Web interface

Some organizations modify the end user workflow in MicroStrategy Web to include custom functionality. Others build custom data transformation logic to render BI information to suit the needs of their applications. Using the MicroStrategy SDK, developers add the desired functionality or alter the report execution flow necessary for their specific applications.

For example, the default execution workflow of a report that contains prompts involves two distinct Web display steps:

- The first page displayed to the user contains the list of prompts in a variety of formats such as radio buttons, check boxes, and drop-down lists. End users make their selections, and click a button to proceed with the report execution. At this point, the prompts are closed and no more prompt selections are possible.
- The second page displays the report results from the report definition and the prompt selections made by the user.

Through changes to the prompt layout and report execution workflow, a customized report execution page can display a prompt pane to the left of the report. This modification can be implemented through simple changes in the page configuration file as well as the associated layout files. Using some Java code, the workflow can be customized so that the report in the right pane is updated based on prompt selections.

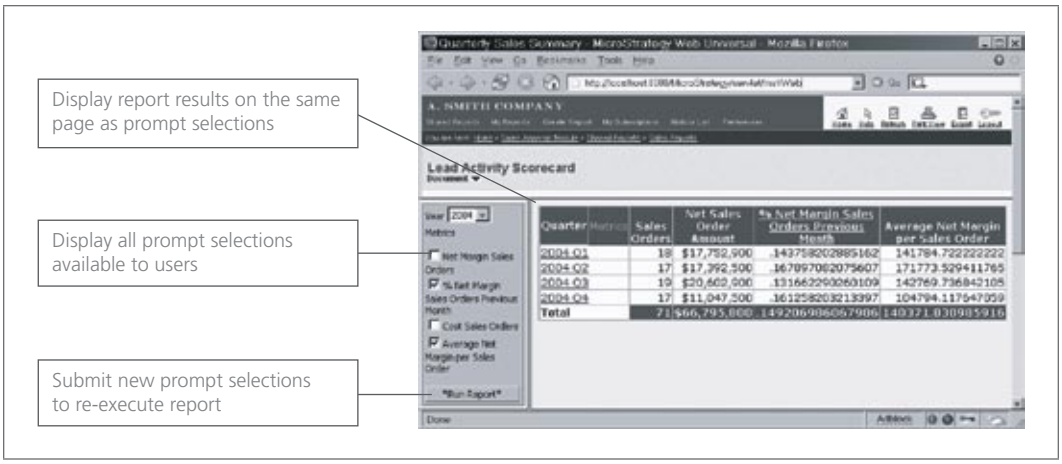


Figure 19.16 Add prompt selection to the report page. Users can change the prompt parameters, and re-execute the report to display the results on the same page.

Integrate with Third-Party Visualization Software

BI applications often need to address the specialized visualization needs of specific audiences. For example, pharmaceutical specialists, and biochemists require the ability to display complex visualizations such as chemical structures of pesticides or molecular structures of genes. Other applications that involve geospatial data require visualizations in map format. For these applications, the MicroStrategy BI platform serves as the data provider, while specialized visualization tools perform the data rendering.

Integration with mapping software such as MapInfo® and ESRI® is a common example of integration with third-party visualization products. Application requirements dictate whether the visualization occurs in a stand-alone window of the visualization software or whether the MicroStrategy Web interface embeds the visualization controls. When a user clicks on a map icon on the toolbar of a report with geospatial data, custom event handlers invoke the mapping software. Add-ons encapsulate the Java code that passes the report data to the mapping software and back to MicroStrategy Web for bidirectional integration. This bidirectional integration allows a seamless completion of the investigative flow, without ever leaving the MicroStrategy Web interface.

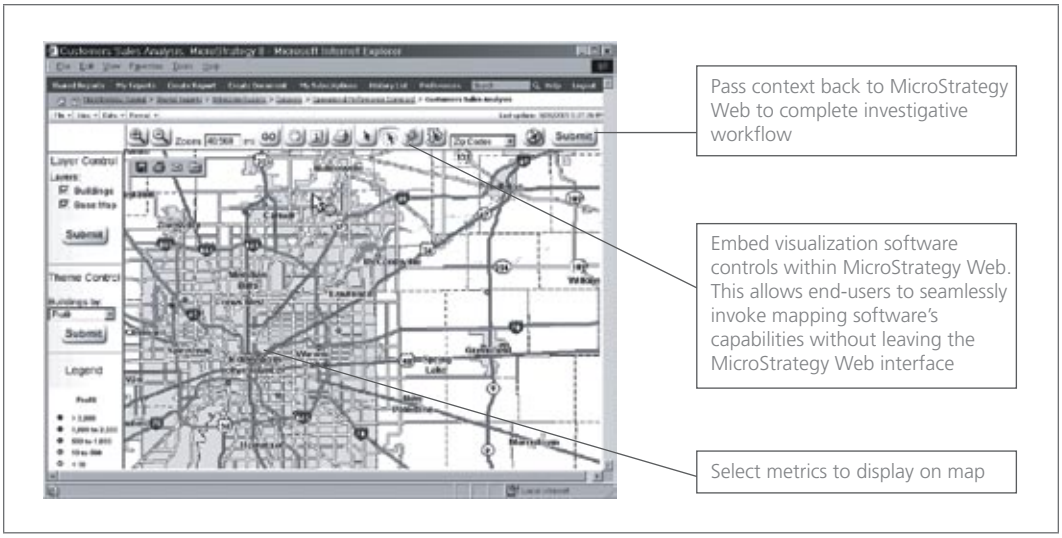


Figure 19.17 Bi-directional integration with a mapping tool allows business users to analyze geographical trends. For example, a marketing analyst can display sales and profitability in a map format to represent the geographic location of high profitability customers.

Implement Single Sign-on Solutions

Organizations today face extensive scrutiny regarding access to sensitive data. To comply with these requirements, they implement sophisticated authentication mechanisms such as biometric authentication and two-factor authentication to verify user credentials. Additionally, organizations require each user to enter his or her credentials when initially accessing the corporate network. Irrespective of how the users are stored in the rest of their enterprise applications, organizations expect every enterprise application to recognize the same user without requiring re-entry of the credentials.

MicroStrategy offers numerous options for implementing Single Sign-On (SSO). An External Security Module, sample code, and comprehensive documentation available in the SDK make it possible to integrate MicroStrategy with portals and identity management systems.

- Single Sign-On within a Portal Deployment: When deployed in an enterprise portal, the MicroStrategy portlet retrieves the user credentials from the credentials vault of the portal server, and passes the credentials to MicroStrategy Web. Organizations also have the flexibility of creating a user session in Intelligence Server prior to executing the user request using the External Security Module.
- Single Sign-On with Identity Management applications: When external authentication is required for all enterprise application users, as in the case of biometric and two factor authentication, the policy server of the identity management system authenticates the user against the user directory. The identity management system passes a security token to MicroStrategy confirming the identity of the user. Custom code implemented using the External Security Module validates the token against the policy server, and authorizes user actions in MicroStrategy Web.

Integration with Office Productivity Software

Another common requirement is that BI applications be accessible from a variety of interfaces. With MicroStrategy Office, business users never need to leave their Microsoft Office application to access MicroStrategy reports. Using the MicroStrategy Office API, organizations create custom applications that incorporate reports from multiple BI applications. These applications automatically use the MicroStrategy Office API to select reports, answer prompts, and embed report results in Microsoft Office. Custom business logic that uses VBA, Visual Basic, Visual Basic.NET, C++, and C# programming is very easy to incorporate on top of the MicroStrategy Office API. As a result, Microsoft Excel, PowerPoint, and Word files become live repositories of scorecards, reports, charts, PivotTables, and Pivot Charts, continually linked back to the MicroStrategy BI Platform for the latest data.

Integration with Other Enterprise Applications

Given the openness of the MicroStrategy BI platform, external applications can instantly take advantage of BI functionality by plugging directly into the platform. The ability to plug into the MicroStrategy platform enables innovative integration with other enterprise applications in several ways:

- Web services provide a cost-effective Enterprise Application Integration (EAI) option. MicroStrategy's open, Service-Oriented Architecture makes it a superior business intelligence platform for Web Service applications. The loosely coupled BI services available using Web Services enable external applications to access any BI application built using the MicroStrategy BI platform. In addition to executing reports in MicroStrategy, external applications can incorporate user selections of prompt answers or report parameters to control the behavior of third-party applications.
- MicroStrategy Narrowcast Server provides an open, documented API that enables organizations to create custom modules for information retrieval from any data source, including non-relational data sources, or information transmission to any new output device. The BI application can integrate with any Web application, database, or file system to retrieve content used in information delivery services. These customized applications can format and transmit data to other applications in specific formats required for those applications.

Integrate with System Management Tools

Automated management and system control is critical to non-stop operations of enterprise-class application deployments. Organizations regularly build sophisticated system control and administration applications or use third-party system management tools to manage their BI infrastructure. Common examples of custom applications include:

- Automatic disconnection and reconnection of database threads: Any third-party Windows application can access Intelligence Servers running on the Windows, Linux, or UNIX operating systems to disconnect all Intelligence Server connections to the database automatically upon the start of the data warehouse load process. The custom programs reconnect the database connections automatically when the data warehouse load is complete.
- Logon metering: Average response times can suffer during periods of unusually heavy application usage. Custom rules can analyze detailed performance statistics and counters in real-time to load management. For instance, under peak loads, additional Web users can be prevented from connecting to the application.
- Assured batch delivery: System management tools use the Delivery Engine API to manage routine operations of the Narrowcast Server Delivery engine. They automatically start and stop the Delivery Engine, connect and disconnect from the distribution manager, query the status of Narrowcast Server components, pause, and execute report delivery services on demand to ensure 24x7 operations with minimal human intervention.

19.13 Summary

The organic service-oriented MicroStrategy BI architecture exposes consistent, encapsulated BI functionality using thousands of documented interfaces and methods. Through documentation of the API, sample code, and utilities, the MicroStrategy SDK empowers developers to implement highly customized, functional, and powerful Web reporting BI applications. Most look and feel customizations of the Web interface are performed using easy to edit and maintain XML configuration files, Cascading Style Sheets and xHTML-like Layout Definition files. No ASP.NET, JSP, or Java coding is required for such presentation-related customizations. More advanced functionality customizations are possible using Java code.

APPENDICES

Appendices

Appendix A: Schema Support

In addition to the modeling techniques described in the chapters, the MicroStrategy BI platform contains features that extend the support for existing data warehouse schemas, from star to snowflake:

- Heterogeneous column mapping
- Business hierarchies
- Attribute forms
- Expressions
- Dimensional extensions
- Logical Tables

For support of operational databases and data marts, the freeform SQL engine supports the following:

- Freeform SQL editor

Support of Multi-dimensional cube databases, such as SAP BW, is achieved through the dynamic MDX engine:

- OLAP Cube Catalog to select cubes for reporting
- Automatic Metadata translation to convert cube metadata to MicroStrategy metadata

Heterogeneous Column Mapping

Data warehouses frequently combine information from disparate source systems. As a result, an attribute represented by a MONTH_ID column in one table might be represented by MNTH_ID in another table, and simply MON in another table. Rather than requiring the batch load process to standardize the column names, MicroStrategy can designate the three column names as being used for the attribute Month. MicroStrategy Intelligence Server will take this equivalency into account when generating SQL and creating joins.

Business Hierarchies

Business analysts often need to pursue drill paths in the course of investigating the data that bear little or no resemblance to the physical warehouse structures. For example, a business analyst may need to drill from a particular store to the promotions that store ran in the past month. Promotions may be stored in a different physical warehouse table than stores, yet MicroStrategy can define a hierarchy that allows the user to seamlessly drill down from Store to Promotion.

Attribute Forms

An attribute such as Month may have several descriptors or attribute forms. For example, Month may be viewed as "January 2002" by some users, "2002 01" to others, "2002 12" to users of a fiscal calendar that ends in January, or "Janvier 2002" to French users. MicroStrategy allows the definition of multiple forms for each attribute that enable each user to see exactly the form that is most relevant to them. Attribute forms even allow the inclusion of HTML image tags in a report, and allow images to be displayed on a report.

Expressions

Ideally, business analysts should not be restricted to only the data that is physically stored in the data warehouse. They should also be able to manipulate particular columns. Taking the month attribute example discussed in heterogeneous column mapping, a month attribute can be defined as "MONTH(date)" using one of MicroStrategy's many built-in functions. Another example is a customer's age that is best calculated by taking the difference between the customer's birth date and the current date, using a database function. Expressions allow such calculations without requiring changes to the ETL process. Expressions can be used to define attributes, facts, transformations, and metrics. These expressions provide unlimited functionality in supporting any schema available.

Fact Allocations

Fact allocations allow a fact to be reported at a level lower than the level at which it is stored. For example, a business unit's budget might be stored at the Quarter level. However, a user might want to report on budget at the Month level, with each month's allocation being one third of the quarter total. Although a new table could be built in the data warehouse to represent this data, that approach adds time to the batch load, and requires additional space in the data warehouse. A fact allocation defined in the metadata using MicroStrategy Architect can calculate a fact expression for Budget at the Month level as "quarterly budget / 3".

Fact Extensions

Fact extensions allow facts to be reported on levels at which they are not stored. For example, the Unit Cost of a product may vary only by product but remain the same across all stores. A report computing the total cost for a specific store must use the Unit Cost fact at the Store level even though the fact is not stored at the store level. A fact extension defined in MicroStrategy Architect allows the unit cost metric to be computed at the store level by extending the unit cost from the product level to the number of specific products sold at that store.

Fact allocations and extensions eliminate unnecessary data storage, and reduce the amount of time needed for data extract, transform, and load processes. Without them, the data warehouse would need to contain additional tables to enable some types of reporting. By defining a fact allocation or extension in MicroStrategy, the SQL for a report can derive the necessary data rather than requiring it in the data warehouse.

Logical Tables

Logical tables allow architects to create data structures different from the underlying table structures in the database by typing in a SQL statement to create a logical table in MicroStrategy. This is similar to the creation of database views, but these logical tables are not created in the database, but are stored in the MicroStrategy metadata.

Logical tables allow architects to deal with complex data modeling issues such as recursive hierarchies which may be flattened, and slowly changing dimensions.

Freeform SQL Editor

Users can create reports against operational databases and other datamarts with data structures not optimal for OLAP reporting by creating SQL statements in a SQL editor. Users can type in any ANSI-compliant SQL statement and even multipass statements if the ODBC driver supports this. Users can reference existing stored procedures, define user prompts, and also insert security filters into these SQL reports.

OLAP Cube Catalog

Designers can create reports against multi-dimensional cube databases such as SAP BW using MicroStrategy's dynamic MDX engine. The OLAP cube catalog allows users to browse the list of available cubes in the cube database, and select the ones that may be exposed to users for use in reporting.

Automatic Metadata Mapping

Once a cube has been selected in the OLAP cube catalog, the MicroStrategy MDX engine automatically translates the cube metadata into corresponding MicroStrategy metadata. For example, SAP characteristics are converted to MicroStrategy attributes, variables to MicroStrategy prompts, and key figures to metrics.

Though the translation process is automatic, developers can edit the names and properties of the translated metadata elements to customize them. Architects can also specify join properties between various data sources. For example, if Customer name information is stored both in SAP BW as well as the data warehouse, architects can specify a common attribute for both. Once a common attribute has been specified, users can join data from these multiple data sources along the conforming dimensions.

Appendix B: Very Large Database (VLDB) Properties

VLDB properties are settings that customize the SQL and MDX generated by MicroStrategy Intelligence Server. The settings provide a way to manipulate SQL join patterns, SQL insert statements, and table creation properties without manually altering SQL Scripts as well as how the Analytical Engine manages certain results. By adjusting the SQL and MDX statements using VLDB properties, specific optimizations can further enhance the performance of queries, and also provide the capability to easily incorporate and take advantage of new RDBMS features introduced in new versions.

VLDB properties can be set at different levels within a MicroStrategy project. The levels are used in the following sequence of decreasing priority:

- Applied only when a specific report is run (Report)
- Every time a template is used with any filtering criteria (Template)
- Whenever processing a metric regardless of its use in reports, filters and/or other metrics (Metric & Transformation)
- For all queries accessing a specific database connection (Database Instance)
- Globally across the project (Project)

The following table lists the VLDB properties that are available to a developer.

Category	Property	Description	Level Available
Analytical Engine	Display Null On Top	Control the display null values at the top or the bottom of a list when sorting data	Report, Template, Project
	Distinguish Duplicate Rows	Control how a duplicated row will be identified and reported	Report, Template, Project
	Null Checking For Analytical Engine	Enable or disable checking for null values in the Analytical Engine	Report, Template, Metric, Project
	Subtotal Dimensionality Aware	Control whether the Analytical Engine takes into account the dimensionality of the subtotal calculation	Report, Template, Metric, Project
	Subtotals Over Consolidations Compatibility	Control the evaluation of consolidations by their elements only or by the elements and the underlying attributes	Project
Governing	Maximum SQL/MDX Size	Maximum size in bytes of the SQL or MDX that can be accepted by the ODBC/BAPI/XMLA driver	Database Instance
	SQL Time Out (Per Pass)	Time out in seconds for each pass of SQL	Report, Database Instance
	Intermediate Row Limit	Set the maximum number of rows allowed in intermediate tables	Report
	Results Set Row Limit	Set the maximum number of rows allowed in report results	Report
Indexing	Index PostString	A string that will be appended at the end of CREATE INDEX statement	Report, Template, Database Instance
	Index Prefix	Prefix to use when automatically creating indexes for intermediate SQL passes	Report, Template, Database Instance
	Index Qualifier	String to insert between the CREATE and INDEX keywords	Report, Template, Database Instance
	Intermediate Table Index	Determine whether and when to create index for intermediate table	Report, Template, Database Instance
	Max Columns in Index	Determine the maximum number of columns that could be included in partition key or index	Report, Template, Database Instance

Category	Property	Description	Level Available
Joins	Cartesian Join Warning	Actions to take when a report contains a Cartesian join.	Report, Template, Database Instance
	Join Type	Set the ANSI standard syntax for joining tables (89 or 92/99)	Report, Template, Database Instance
	Preserve All Lookup Table Elements	Determine which columns should be preserved when combining the intermediate SQL passes	Report, Template, Database Instance
Metrics	Integer Constant In Metric	Apply a decimal to integer values that are used in calculations to prevent datatype conversions of decimals to integers	Metric, Database Instance
	Metric Join Type	Define whether to use inner or outer joins when evaluating a metric	Metric, Database Instance
	Null Check	Indicates how to handle null values in SQL statements	Report, Template, Metric, Database Instance
	Zero Check	Indicates how to handle division by zero checking in SQL statements	Report, Template, Metric, Database Instance
	Transformation Role Processing	Choose whether to apply transformation processing to all attributes or only to highest level child attribute applicable to multiple attributes	Transformation
	Metric Column Alias	Define an alias to use for a metric in SQL statements	Metric
	Data Type	Define the data type to use for the metric in SQL statements	Metric
Pre/Post SQL Statements[1]	Cleanup Post Statement	Adds a SQL statement/comment at the end of the MicroStrategy generated SQL code	Report, Template, Database Instance
	Insert Mid Statement	Adds a SQL statement/comment after each individual insert statement	Report, Template, Database Instance
	Insert Post Statement	Adds a SQL statement/comment after the final insert statement	Report, Template, Database Instance
	Insert Pre Statement	Adds a SQL statement/comment before the final insert statement	Report, Template, Database Instance
	Report Post Statement	Adds a SQL statement/comment prior to dropping the first intermediate table	Report, Template, Database Instance
	Report Pre Statement	Adds a SQL statement/comment prior to creating the first intermediate table	Report, Template, Database Instance
	Table Post Statement	Adds a SQL statement/comment after creating each intermediate table	Report, Template, Database Instance
	Table Pre Statement	Adds a SQL statement/comment prior to creating each intermediate table	Report, Template, Database Instance
Query Optimizations	Apply Filter Options	Determine whether to apply filter conditions to intermediate SQL statements and/or the final SQL statement	Report, Template, Database Instance
	MD Partition PreQuery Option	Choose whether to use a COUNT or a constant when initially querying partition tables	Report, Template, Database Instance
	Rank Method If DB Ranking Not Used	Define whether ranking calculations are performed by ODBC ranking or Analytical Engine if the database doesn't support ranking	Report, Template, Database Instance
	Sub Query Type	Control if EXISTS, IN or intermediate tables are used when performing subqueries	Report, Template, Database Instance
	WHERE Clause Driving Table	Determine whether to apply filters to the lookup tables or to the fact tables	Report, Template, Database Instance

[1] There are five separate statements for each of the Pre/Post SQL statements. For the first four statements, each contains a single SQL stateme
Pre/Post SQL Statements.

Category	Property	Description	Level Available
Select/Insert	Attribute Form Selection Option For Intermediate Pass	Choose to select only the Attribute ID or all Attribute forms on the report in intermediate SQL passes	Report, Template, Database Instance
	Attribute Selection Option For Intermediate Pass	Choose to select the Attribute ID at the level of the report or all other Attribute ID levels in intermediate SQL passes	Report, Template, Database Instance
	Distinct/Group By Option (when no aggregation and not table key)	Choose whether to use DISTINCT, GROUP BY or none of these when selecting information that is not being aggregated	Report, Template, Database Instance
	GROUP BY Non-ID Attribute	Choose whether to use MAX or GROUP BY when retrieving non-ID Attribute forms	Report, Template, Database Instance
	Insert Post String	Adds a SQL string to the end of INSERT statements when populating intermediate tables	Report, Template, Database Instance
	Insert Table Option	Adds a SQL string immediately preceding the SELECT statement of INSERT statements when populating intermediate tables	Report, Template, Database Instance
	Merge Same Metric Expression Option	Choose whether to select a metric more than once if it is used more than once in a report	Report, Template, Database Instance
	Select Post String	Adds a SQL string to the end of the SELECT statements in the final SQL pass	Report, Template, Database Instance
	SQL Date Format	Enter the mask used for Date datatypes in SQL statements	Database Instance
	SQL Decimal Separator	Choose whether to use a period or comma as the decimal separator in SQL statements	Report, Template, Database Instance
	SQL Hint	Adds a SQL hint immediately after the word, SELECT, in all SQL passes	Report, Template, Database Instance
	SQL Timestamp Format	Enter the mask used for Timestamp datatypes in SQL statements	Database Instance
	UNION Multiple INSERT	Choose whether it use UNION to populate an intermediate table or to use more than one INSERT statement	Report, Template, Database Instance
Tables	Attribute ID Constraint	Adds a SQL string to ID columns in intermediate tables	Database Instance
	Commit After Final Drop	Define whether to issue COMMIT SQL statement after the final intermediate table has been dropped	Report, Database Instance
	Commit Level	Define whether to issue a COMMIT SQL statement after DDL instructions, after DML instructions, after both DDL and DML instructions or never	Report, Template, Database Instance
	Create Post String	Adds a SQL string to the end of the CREATE TABLE statements in intermediate SQL passes	Report, Template, Database Instance
	Drop Temp Table Method	Define whether to clean up the intermediate tables immediately or not	Report, Template, Database Instance
	Fallback Table Type	Choose whether to use true temporary tables or permanent tables for fallback tables	Report, Template, Database Instance
	Intermediate Table Type	Choose whether to use true temporary tables, derived tables, common table expressions, temporary views or permanent tables for intermediate tables	Report, Template, Database Instance
	Table Creation Type	Define whether intermediate tables are created implicitly in a single SQL pass or explicitly in two SQL passes	Report, Template, Database Instance
	Table Descriptor	Adds a SQL string immediately after the words CREATE TABLE in intermediate SQL passes	Report, Template, Database Instance
	Table Option	Adds a SQL string immediately after the table name in the CREATE TABLE statements in intermediate SQL passes	Report, Template, Database Instance
	Table Prefix	Adds a prefix to the table name in the CREATE TABLE statements in intermediate SQL passes	Report, Template, Database Instance
	Table Qualifier	Adds a SQL string between the words CREATE and TABLE in intermediate SQL passes	Report, Template, Database Instance
	Table Space	Adds a SQL string immediately after the column list in the CREATE TABLE statements in intermediate SQL passes	Report, Template, Database Instance

Category	Property	Description	Level Available
Report Data Options	Report Limit	Applies additional filtering conditions to metrics after calculation of the metrics	Report, Template
	Metric Join Type	Define which metrics should use an inner or outer join against the other metrics on a template	Report, Template
	Attribute Join Type	Define which attributes should use an inner or outer join against the other attributes on a template based on SQL passes and filtering criteria	Report, Template
	Evaluation Order	Determine the sequence in which calculations in Analytical Engine should be evaluated	Report, Template
MDX Settings			
Analytical Engine	Display Null On Top	Control the display null values at the top or the bottom of a list when sorting data	Report, Project
	Distinguish Duplicate Rows	Control how a duplicated row will be identified and reported	Report, Project
	Null Checking For Analytical Engine	Enable or disable checking for null values in the Analytical Engine	Report, Project
Governing	Results Set Row Limit	Set the maximum number of rows allowed in report results	Report, Project
Select/Insert	SQL Decimal Separator	Choose whether to use a period or comma as the decimal separator in SQL statements	Report, Project
	Add NON EMPTY	Set whether or not to add NON EMPTY to the MDX SELECT statement	Report, Project

Appendix C: Analytical Functions

Basic Functions
Average
Count
First
Geometric Mean
Greatest
Last
Least
Maximum
Median
Minimum
Mode
Product
Standard Deviation of Population
Standard Deviation of Sample
Sum
Variance of Population
Variance of Sample

Date and Time Functions
Add Days to a Date
Add Month to a Date
Current Date
Current Datetime
Current Time
Date
Day of Month
Day of Week
Day of Year
Days Between
Hour
Millisecond
Minute
Month
Month End Date
Month Start Date
Months Between
Quarter
Second
Week
Year
Year End Date
Year Start Date

Internal Functions
Apply Aggregation
Apply Comparison
Apply Freeform SQL Security Filter
Apply Logic
Apply OLAP
Apply Optional
Apply Simple
Banding based on Size
Banding based on Count
Banding based on Percent
Case
Case Value
Intersect
Intersect In
Tuple

Null/Zero Functions
Is Not Null
Is Null
Null To Zero
Zero To Null

OLAP Functions
Exponential Weight Moving Average
Exponential Weight Running Average
First Value in Range
Last Value in Range
Moving Average
Moving Count
Moving Difference
Moving Maximum
Moving Minimum
Moving Standard Deviation of Population
Moving Standard Deviation of Sample
Moving Sum
Running Average
Running Count
Running Maximum
Running Minimum
Running Total
Running Standard Deviation of Population
Running Standard Deviation of Sample
Running Sum
Rank and NTile Functions
N-Tile
N-tile by Step
N-tile by Value
N-tile by Step and Value
Percentile
Rank

String Functions
Concatenate
Concatenate with a space
Initial Capitalization
Left String
Length
Lower
Left Trim
Position
Right String
Right Trim
SubString
Trim
Upper

Mathematical Functions
Absolute
Arc cosine
Arc cosine hyperbolic
Arc sine
Arc sine hyperbolic
Arc tangent
Arc tangent2
Arc tangent hyperbolic
Arc tangent hyperbolic
Ceiling
Combine
Cosine
Cosine hyperbolic
Degrees
Exponent
Factorial
Floor
Integer
Log
Log Base 10
Modulus
Natural Log
Power
Quotient
Radians
Random Number Between
Round
Round with Precision
Sine
Sine hyperbolic
Square Root
Tangent
Tangent hyperbolic
Truncate

Statistical Functions
Average Deviation
Beta Distribution
Binomial Distribution
Chi-Square Distribution
Chi-Square Test
Confidence Interval
Correlation Coefficient
Covariance
Criterion Binomial Distribution
Exponential Distribution
Fisher Transformation
F-Probability Distribution
F-Test
Gamma Distribution
Heteroscedastic Ttest
Homoscedastic Ttest
Hypergeometric Distribution
Intercept
Inverse of Beta Distribution
Inverse of Chi-Square Distribution
Inverse of F Probability Distribution
Inverse of Fisher Transformation
Inverse of Gamma Distribution
Inverse of Lognormal Cumulative Distribution
Inverse of the Normal Cumulative Distribution
Inverse of the Standard Normal Cumulative Standard
Inverse of T-Distribution
Kurtosis
Lognormal Cumulative Distribution
Mean
Mean T-Test
Negative Binomial Distribution
Normal Cumulative Distribution
Paired T-test
Pearson Product Moment Correlation Coefficient
Permutation
Poisson Distribution
RSquare
Skew
Slope of Linear Regression
Standardize
Standard Normal Cumulative Distribution
Standard Error of Estimates
T-Distribution
Variance Test
Weibull Distribution

Financial Functions
Accrued Interest
Accrued Interest Maturity
Amount Received at Maturity
Convert Dollar Price from Decimal to Fraction
Convert Dollar Price from Fraction to Decimal
Coupon Period, Days Beginning to Settlement
Coupon Period, Days with Settlement
Coupon Period, Days from Settlement to Next Coupon
Cumulative Interest Paid on Loan
Cumulative Principal Paid on Loan

Discount Rate for a Security
Double-Declining Balance Method
Effective Annual Interest Rate
Fixed-Declining Balance Method
Future Value
Future Value Schedule
Interest Rate
Interest Payment
Internal Rate of Return
Interest Rate per Period
Macaulay Duration
Modified Duration
Modified Internal Rate of Return
Net Present Value
Next Coupon Date After Settlement Date
Nominal Annual Interest Rate
No. of Coupons Between Settlement and Maturity
Number of Investment Periods
Odd First period Price
Odd First period Yield
Odd Last Period Price
Odd Last Period Yield
Payment
Payment Principal
Previous Coupon Date Before Settlement Date
Price Per $100 Face Value
Price, Discounted
Price at Maturity
Present Value
Straight Line Depreciation
Sum-Of-Years' Digits Depreciation
Treasury Bill Equity
Treasury Bill Price
Treasury Bill Yield
Variable Declining Balance
Yield
Yield for Discounted Security
Yield at Maturity

Data Mining Functions
Clustering, Numeric
Clustering, Non-Numeric
General Regression, Numeric
General Regression, Non-Numeric
Mining Model, Numeric
Mining Model, Non-Numeric
Neural Network, Numeric
Neural Network, Non-Numeric
Regression, Numeric
Regression, Non-Numeric
Train Regression Model
Train Regression Model with Tree
Tree Model, Numeric
Tree Model, Non-Numeric

Arithmetic Operators
Addition
Division
Multiplication
Subtraction
Unary Minus

Comparison Operators
Begins With
Between
Contains
Ends With
Equal
Greater Than
Greater Than or Equal
In
Less Than
Less Than or Equal
Like
Not Begins With
Not Between
Not Contains
Not Ends With
Not Equal
Not In
Not Like

Comparison for Rank Operators
Between
Equal
Greater Than or Equal
Less Than or Equal
Not Between
Not Equal

Logical Operators
And
If
Not
Or

Appendix D: Supported Graph Types

Area – Horizontal absolute
Area – Horizontal stacked
Area – Horizontal bipolar absolute
Area – Horizontal bipolar stacked
Area – Horizontal dual axis absolute
Area – Horizontal dual axis stacked
Area – Horizontal percent
Area – Vertical absolute
Area – Vertical stacked
Area – Vertical bipolar
Area – Vertical bipolar stacked
Area – Vertical dual axis absolute
Area – Vertical dual axis stacked
Area – Vertical percent
Area –3D group
Area –3D series
Bar – Horizontal side by side
Bar – Horizontal stacked
Bar – Horizontal dual axis side by side
Bar – Horizontal dual axis stacked
Bar – Horizontal bipolar side by side
Bar – Horizontal bipolar stacked
Bar – Horizontal percent
Bar – Horizontal absolute
Bar – Vertical side by side
Bar – Vertical stacked
Bar – Vertical dual axis side by side
Bar – Vertical dual axis stacked
Bar – Vertical bipolar side by side
Bar – Vertical bipolar stacked
Bar – Vertical percent
Bar – Vertical absolute
Bar – 3D rectangular riser

Bar – 3D columnar riser
Bar – 3D conical riser
Bar – 3D floating cube
Bar – 3D floating sphere
Line – Horizontal absolute
Line – Horizontal stacked
Line – Horizontal bipolar absolute
Line – Horizontal bipolar stacked
Line – Horizontal dual axis absolute
Line – Horizontal dual axis stacked
Line – Horizontal percent
Line – Vertical absolute
Line – Vertical stacked
Line – Vertical bipolar absolute
Line – Vertical bipolar stacked
Line – Vertical dual axis absolute
Line – Vertical dual axis stacked
Line – Vertical percent
Line – 3D group
Line – 3D series
Pie – Standard
Pie – Ring
Pie – Multiple
Pie – Multiple ring
Pie – Multiple proportional
Pie – Multiple proportional ring
Pie – 3D basic
Pie – 3D ring
Pie – 3D multiple
Pie – 3D multiple ring
Pie – 3D multiple proportional
Pie – 3D multiple proportional ring
Scatter–X-Y

Scatter r– X-Y
Scatter – Dual axis X-Y
Scatter – X-Y-Z
Polar – Coordinate
Polar – Dual axis coordinate
Radar – Line
Radar – Stacked line
Radar – Dual axis line
Radar – Dual axis stacked line
Radar – Area
Radar – Stacked area
Bubble – Standard
Bubble – Dual axis
Stock – Hi Lo
Stock – Hi Lo Close
Stock – Hi Lo open close
Histogram – Horizontal
Histogram – Vertical
3D Surface – Standard
3D Surface – Standard with sides
3D Surface – Honeycomb
Gauge – Standard
Funnel – Standard
Pareto – Standard
Pareto – Percent
BoxPlot – Box and whiskers
Gantt – Standard
Combination – Bar and area
Combination – Bar and line
Combination – Area and line
Combination – Dual axis bar and area
Combination – Dual axis bar and line
Combination – dual axis area and line

Appendix E: Performance Counters and Key Performance Indicators

MicroStrategy Intelligence Server Operations Counters
Average report queue time
Average report response time
Element browse submission rate
Largest free region size
Number of executing reports
Number of timed-out jobs
Object browse submission rate
Open reports
Report cache hit ratio
Report submission rate
Report completion rate
Total completed documents
Total completed reports
Total completed Web reports
Total completed element browse requests
Total completed object browser requests
Total documents
Total element browse requests
Total error documents
Total error element browse requests
Total error object browser requests
Total error reports
Total MCM denials
Total object browse requests
Total report cache checks
Total report cache hits
Total report requests
Total requests

MicroStrategy Intelligence Server User Statistics
Average user session length
Number of invalid login attempts
Open project sessions
Open sessions
Total project sessions
Total sessions
Total timed out users

MicroStrategy Intelligence Server Process Statistics (Windows)
% privileged time
% processor time
% user time
Creating process ID
Elapsed time
Handle count
ID process
I/O data bytes/sec
I/O data operations/sec
I/O other bytes/sec
I/O other operations/sec
I/O read bytes/sec
I/O read operations/sec
I/O write bytes/sec
I/O write operations/sec
Page faults/sec
Page file bytes
Page file bytes peak
Pool non-paged bytes
Pool paged bytes
Priority base
Private bytes
Thread count
Virtual bytes
Virtual bytes peak
Working set
Working set peak

Enterprise Manager Key Performance Indicators
Average Connection Duration hh:mm:ss
Average Connection Duration per User hh:mm:ss
Average Number of Active Sessions
Average Number of Active Users
Connect Date
Connection Duration hh:mm:ss
Day Date - Prompted
DP Average Elapsed Duration per Job hh:mm:ss
DP Average Execution Duration per Job hh:mm:ss
DP Average Number of Jobs per Session
DP Average Number of Prompts
DP Average Number of Reports per Document
DP Average Queue Duration per Job hh:mm:ss
DP Elapsed Duration hh:mm:ss
DP Execution Duration hh:mm:ss
DP First Exec Finish Timestamp
DP First Exec Request Timestamp
DP First Exec Start Timestamp
DP Last Exec Finish Timestamp
DP Last Exec Request Timestamp
DP Last Exec Start Timestamp
DP Number of Jobs
DP Number of Jobs w/o Error
DP Number of Jobs with Error
DP Queue Duration hh:mm:ss
First Connect Timestamp
First Disconnect Timestamp
Last Connect Timestamp
Last Disconnect Timestamp
Max Connection Duration hh:mm:ss
Max Number of Active Sessions
Max Number of Active Users
Min Connection Duration hh:mm:ss
Min Number of Active Sessions
Min Number of Active Users
Number of Attribute Forms
Number of Attributes
Number of Closed Sessions
Number of Columns
Number of Consolidations
Number of Continuous Connected Users
Number of Continuous Sessions
Number of Custom Groups
Number of Days
Number of Days - Document JobsNumber of Days - Report Jobs
Number of DB Instances
Number of Disconnected Users
Number of Documents
Number of Encapsulated Connected Users
Number of Encapsulated Sessions
Number of Events
Number of Facts
Number of Filters
Number of Hierarchies
Number of Intelligence Servers
Number of Metrics
Number of New Connected Users
Number of New Opened Sessions
Number of Projects 1
Number of Prompts
Number of Reports
Number of Schedules
Number of Sessions
Number of Tables
Number of Templates
Number of Transformations

Number of Users
Number of Users (no group-by Week of Year)
Number of Web Sessions
Number of Web Users
RP Activity Analysis Metric
RP Average Analytical Engine Exec Duration hh:mm:ss
RP Average Analytical Engine Exec Duration w/o Queue Time hh:mm:ss_deprecated
RP Average CPU Duration per Job
RP Average Daily Use Duration per job hh:mm:ss
RP Average Elapsed Duration per Job hh:mm:ss
RP Average Elapsed Duration per User hh:mm:ss
RP Average Elapsed Duration per Web Job hh:mm:ss
RP Average Exec Duration per Job hh:mm:ss
RP Average Number of Ad-Hoc Jobs per Report
RP Average Number of Ad-Hoc Jobs per Session
RP Average Number of Ad-Hoc Jobs per User
RP Average Number of Cancelled Jobs per Report
RP Average Number of Cancelled Jobs per Session
RP Average Number of Cancelled Jobs per User
RP Average Number of DB Result Rows per Job
RP Average Number of DB Result Rows per Session
RP Average Number of DB Result Rows per User
RP Average Number of DB SQL Passes per Job
RP Average Number of Document Jobs per Report
RP Average Number of Document Jobs per Session
RP Average Number of Document Jobs per User
RP Average Number of Drill Jobs per Report
RP Average Number of Drill Jobs per Session
RP Average Number of Drill Jobs per User
RP Average Number of Jobs per Report
RP Average Number of Jobs per Session
RP Average Number of Jobs per User
RP Average Number of Jobs per Web User
RP Average Number of Jobs w/o Cache Creation per Report
RP Average Number of Jobs w/o Cache Creation per Session
RP Average Number of Jobs w/o Cache Creation per User
RP Average Number of Jobs w/o Cache Hit per Report
RP Average Number of Jobs w/o Cache Hit per Session
RP Average Number of Jobs w/o Cache Hit per User
RP Average Number of Jobs w/o Datamart Creation per Report
RP Average Number of Jobs w/o Datamart Creation per Session
RP Average Number of Jobs w/o Datamart Creation per User
RP Average Number of Jobs w/o DB Error per Report
RP Average Number of Jobs w/o DB Error per Session
RP Average Number of Jobs w/o DB Error per User
RP Average Number of Jobs w/o Element Loading per Report
RP Average Number of Jobs w/o Element Loading per Session
RP Average Number of Jobs w/o Element Loading per User
RP Average Number of Jobs w/o Error per Report
RP Average Number of Jobs w/o Error per Session
RP Average Number of Jobs w/o Error per User
RP Average Number of Jobs w/o Security Filter per Report
RP Average Number of Jobs w/o Security Filter per Session
RP Average Number of Jobs w/o Security Filter per User
RP Average Number of Jobs with Cache Creation per Report
RP Average Number of Jobs with Cache Creation per Session
RP Average Number of Jobs with Cache Creation per User
RP Average Number of Jobs with Cache Hit per Report
RP Average Number of Jobs with Cache Hit per Session
RP Average Number of Jobs with Cache Hit per User
RP Average Number of Jobs with Datamart Creation per Report
RP Average Number of Jobs with Datamart Creation per Session
RP Average Number of Jobs with Datamart Creation per User
RP Average Number of Jobs with DB Error per Report
RP Average Number of Jobs with DB Error per Session
RP Average Number of Jobs with DB Error per User
RP Average Number of Jobs with Element Loading per Report
RP Average Number of Jobs with Element Loading per Session
RP Average Number of Jobs with Element Loading per User
RP Average Number of Jobs with Error per Report
RP Average Number of Jobs with Error per Session
RP Average Number of Jobs with Error per User
RP Average Number of Jobs with Security Filter per Report
RP Average Number of Jobs with Security Filter per Session
RP Average Number of Jobs with Security Filter per User
RP Average Number of Non-Ad-Hoc Jobs per Report
RP Average Number of Non-Ad-Hoc Jobs per Session
RP Average Number of Non-Ad-Hoc Jobs per User
RP Average Number of Non-Cancelled Jobs per Report
RP Average Number of Non-Cancelled Jobs per Session
RP Average Number of Non-Cancelled Jobs per User
RP Average Number of Non-Document Jobs per Report
RP Average Number of Non-Document Jobs per Session
RP Average Number of Non-Document Jobs per User
RP Average Number of Non-Drill Jobs per Report
RP Average Number of Non-Drill Jobs per Session
RP Average Number of Non-Drill Jobs per User
RP Average Number of Non-Prompted Jobs per Report
RP Average Number of Non-Prompted Jobs per Session
RP Average Number of Non-Prompted Jobs per User
RP Average Number of Non-Scheduled Jobs per Report
RP Average Number of Non-Scheduled Jobs per Session
RP Average Number of Non-Scheduled Jobs per User
RP Average Number of Prompted Jobs per Report
RP Average Number of Prompted Jobs per SessionRP Average Number of Prompted Jobs per User
RP Average Number of Scheduled Jobs per Report
RP Average Number of Scheduled Jobs per Session
RP Average Number of Scheduled Jobs per User
RP Average Number of SQL Passes per Session
RP Average Number of SQL Passes per User
RP Average Queue Duration per Job hh:mm:ss
RP Average SQL Execution Duration hh:mm:ss
RP Average SQL Execution Duration w/o Queue Time hh:mm:ss_deprecated
RP Average SQL Generation Duration hh:mm:ss
RP Average SQL Generation Duration w/o Queue Time hh:mm:ss_deprecated
RP Cancelled Jobs with SQL exec
RP Cancelled Jobs without SQL exec
RP Count PU Jobs
RP CPU Duration (msec)
RP Elapsed Duration by Non-Web Users hh:mm:ss
RP Elapsed Duration by Web Users hh:mm:ss
RP Elapsed Duration hh:mm:ss
RP Exec Duration hh:mm:ss
RP Export Engine Jobs
RP First Exec Finish Timestamp
RP First Exec Request Timestamp
RP First Exec Start Timestamp
RP First Job Request Timestamp
RP Jobs with No Data Returned
RP Last Exec Finish Timestamp
RP Last Exec Request Timestamp
RP Last Exec Start Timestamp
RP Last Job Request Timestamp
RP Max CPU Duration per Job (msec)
RP Max Elapsed Duration per Job hh:mm:ss
RP Max Exec Duration per Job hh:mm:ss
RP Max Number of DB Result Rows per Job
RP Max Number of DB Tables Accessed per Job
RP Max Number of SQL Passes per Job
RP Max Queue Duration per Job hh:mm:ss
RP Median(Activity Analysis Metrics)
RP Min CPU Duration per Job
RP Min Elapsed Duration per Job hh:mm:ss
RP Min Exec Duration per Job hh:mm:ss
RP Min Number of DB Result Rows per Job
RP Min Number of DB Tables Accessed per Job
RP Min Number of SQL Passes per Job
RP Min Queue Duration per Job hh:mm:ss

RP Number of Ad-Hoc Jobs
RP Number of Cancelled Jobs
RP Number of DB Result Rows
RP Number of DB Tables Accessed
RP Number of Drill Jobs
RP Number of Jobs
RP Number of Jobs accessing DB table/column
RP Number of Jobs Today
RP Number of Jobs w/o Cache Creation
RP Number of Jobs w/o Cache Hit
RP Number of Jobs w/o Datamart Creation
RP Number of Jobs w/o DB Error
RP Number of Jobs w/o Element Loading
RP Number of Jobs w/o Error
RP Number of Jobs w/o Security Filter
RP Number of Jobs with Cache Creation
RP Number of Jobs with Cache Hit
RP Number of Jobs with Datamart Creation
RP Number of Jobs with DB Error
RP Number of Jobs with Element Loading
RP Number of Jobs with Error
RP Number of Jobs with Security Filter
RP number of Narrowcast Server jobs
RP Number of Non-Ad-Hoc Jobs
RP Number of Non-Cancelled Jobs
RP Number of Non-Drill Jobs
RP Number of Non-Prompted Jobs
RP Number of Non-Scheduled Jobs
RP Number of Prompted Jobs
RP Number of Prompts
RP Number of Report Jobs exclusively from Report Execution
RP Number of Report Jobs from Document Execution
RP Number of Reports Used
RP Number of Reports Used (at Project Level)
RP Number of Scheduled Jobs
RP Number of SQL Passes
RP Number of Web Jobs
RP Number of Web Jobs w/o Cache Hit
RP Number of Web Jobs with Cache Hit
RP Percent of Ad-Hoc Jobs
RP Percentage of Cancelled Jobs
RP Percentage of Drill Jobs
RP Percentage of Jobs w/o Cache Creation
RP Percentage of Jobs w/o Cache Hit
RP Percentage of Jobs w/o Datamart Creation
RP Percentage of Jobs w/o DB Error
PR Percentage of Jobs w/o Element Loading
RP Percentage of Jobs w/o Error
RP Percentage of Jobs w/o Security Filter
RP Percentage of Jobs with Cache Creation
RP Percentage of Jobs with Cache Hit
RP Percentage of Jobs with Datamart Creation
RP Percentage of Jobs with DB Error
RP Percentage of Jobs with Element Loading
RP Percentage of Jobs with Error
RP Percentage of Jobs with Security Filter
RP Percentage of Narrowcast Server jobs
RP Percentage of Non-Ad-Hoc Jobs
RP Percentage of Non-Cancelled Jobs
RP Percentage of Non-Drill Jobs
RP Percentage of Non-Prompted Jobs
RP Percentage of Non-Scheduled Jobs
RP Percentage of Prompted Jobs
RP Percentage of Report Jobs exclusively from Report Execution
RP Percentage of Report Jobs from Document Execution
RP Percentage of Scheduled Jobs
RP Percentage Usage by Non-Web Users
RP Percentage Usage by Web Users
RP Queue Duration hh:mm:ss
RP Request Date
RP RunningSum(Activity Analysis Metrics)
RP Schedule Relations
RP SQL Size
RP Timed out jobs
RP User Report Requests
RP Waiting for AutoPrompt Time
RP Waiting for Execution Time
Successful Document Jobs < 1 min
Successful Document Jobs > 5 min
Successful Document Jobs 1-5 min
Successful Jobs < 1 min
Successful Jobs > 5 min
Successful Jobs 1-5 min
Total Number of Application Objects
Total Number of Configuration Objects
Total Number of Schema Object

Appendix F: Supported Data Sources

Data Source	Windows Environment	UNIX/Linux Environment
Generic ANSI SQL92 database	Data Warehouse	Data Warehouse
IBM® DB2® UDB for OS/390 7.1	Data Warehouse Metadata Repository	Data Warehouse Metadata Repository
IBM DB2 UDB for z/OS 8.1	Data Warehouse Metadata Repository	Data Warehouse Metadata Repository
IBM DB2 UDB for iSeries V5R2	Data Warehouse	Data Warehouse
IBM DB2 UDB for iSeries V5R3	Data Warehouse	Data Warehouse
IBM DB2 UDB 8.1 for Windows & UNIX	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
IBM DB2 UDB 8.2 for Windows & UNIX	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
IBM® Information Integrator® 8.1	Data Warehouse	
Informix® Dynamic Server 9.3	Data Warehouse Metadata Repository	Data Warehouse Metadata Repository
Informix Dynamic Server 9.4	Data Warehouse Metadata Repository	Data Warehouse Metadata Repository
Informix Dynamic Server 10	Data Warehouse Metadata Repository	Data Warehouse Metadata Repository
Informix Extended Parallel Server 8.4	Data Warehouse	Data Warehouse
Informix Extended Parallel Server 8.5	Data Warehouse	Data Warehouse
Microsoft Access 2000	Data Warehouse Metadata Repository	
Microsoft Access 2002	Data Warehouse Metadata Repository	
Microsoft Access 2003	Data Warehouse Metadata Repository	
Microsoft Excel 2000	Data Warehouse	
Microsoft Excel 2003	Data Warehouse	
Microsoft SQL Server 7.0 SP4	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging
Microsoft SQL Server 2000	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging
Netezza® 2.2	Data Warehouse	Data Warehouse
Netezza 2.5	Data Warehouse	Data Warehouse
Oracle 8.1.7	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
Oracle 9i	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
Oracle 9iR2	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
Oracle 10g	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging

Data Souce	Windows Environment	UNIX/Linux Environment
Oracle Transparent Gateway 9i	Data Warehouse	
Red Brick® 6.2	Data Warehouse	Data Warehouse
Red Brick 6.3	Data Warehouse	Data Warehouse
SAP BW 3.1	Data Warehouse	Data Warehouse
SAP BW 3.5	Data Warehouse	Data Warehouse
Sybase® Adaptive Server® 12.5	Data Warehouse Metadata Repository	Data Warehouse Metadata Repository
Sybase IQ 12.4.3	Data Warehouse	Data Warehouse
Sybase IQ 12.5	Data Warehouse	Data Warehouse
Sybase IQ 12.6	Data Warehouse	Data Warehouse
Tandem SQL/MP	Data Warehouse	
Teradata® V2R5	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
Teradata V2R5.1	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
Teradata V2R6	Data Warehouse Metadata Repository Narrowcast Server Repository Statistics Logging	Data Warehouse Metadata Repository Statistics Logging
Text Files	Data Warehouse	

Appendix G: Security Privileges

The table below lists all the security privileges that govern access to the BI functionality in the MicroStrategy BI platform. These can be granted by user group, security role, or individual user.

End-user Functionality	Development Power	Administration Control
Run Report Services documents	Create Report Services documents	Create schedules
Change preferences	Create interactive reports	Manage clusters
Change grid/graph view	Format grid properties	Manage caches
Create email address	Format graph properties	Manage database connections
Drill up and down	Design Intelligent Cubes	Create database instances
Search	Create autostyles	Monitor jobs
Print	Modify report data options	Configure projects
Rerun report on the data source	Use data explorer	Manage schedules
Schedule email delivery	Create thresholds	Monitoring user connections
Schedule printer delivery	Create consolidations	Manage users
Subscribe to reports	Create custom groups	Configure MicroStrategy Web
Send report via email	Create attributes	Manage security
Sort	Create hierarchies	Use Command Manager
Use page-by	Create facts	Use Object Manager
View history list	Create metrics	Configure Intelligence Server
Add/remove objects to/from report	Create templates	
Create and format derived metrics	Create filters	
Show Intelligent Cube objects	Use find and replace	
Create view filter	Create cube report	
Add report to history list	Create subtotals	
Drill anywhere	Create prompts	
Define column aliases	Change VLDB properties	
Choose attributes form	Create Freeform SQL report	
Configure toolbars	Create custom drill maps	
Define deliver file location	Create transformations	
Save ad hoc analysis	Create data mining metric	
Define delivery printer	Create project documentation	
Drill on metrics	Create new project	
Export very large file	Import OLAP cube structure	
Run datamart report	Import data warehouse structure	
Export report		
Filter by selection		
Manage objects		
Add subtotals		
Pivot		
View report Details		
View report SQL/MDX		
Save to personal folder		
Save to shared folders		
Schedule file delivery		
Format graph (simple)		
Asynchronous report execution		
Enable header locking		
Drill in Intelligent Cube		
Create folder		
Use report caches		
Run interactive reports		
Use MicroStrategy Office		
Enable Web online training		

Appendix H: API Classes, Methods, Properties, and Interfaces

MICROSTRATEGY WEB API

AbstractAdminTransform:
AbstractAppAddOn:
"AbstractAppComponent
AbstractAppTransform.Context:
AbstractAppTransform:
AbstractAppWebFeatures:
AbstractBodyTag:
AbstractBodyTagHelper:
AbstractCategories:
AbstractCategory:
AbstractChangePasswordTransform:
AbstractConfigurableContainerServices:
AbstractConfigurableParameterBuilder:
AbstractConfigurationElement:
AbstractConfigurationElementList:
AbstractCounter:
AbstractDiagnosticsTransform:
AbstractDocumentTransform:
AbstractEditorTransform:
AbstractEventHandlerFactory:
AbstractExpressionTransform:
"AbstractExternalSecurity:
AbstractFilterElementTransform:
AbstractFilterOperatorContextMenuItem Builder:
AbstractFindDisplayUnitsVisitor:
AbstractFolderContextMenuItemBuilder:
AbstractFolderTransform.
AbstractFrameEditorBean:
AbstractGuiElement:
AbstractIfTag:
AbstractIfTagHelper:
AbstractInstance:
AbstractLayoutSource.LayoutCacheHint:
"AbstractLayoutSource:
AbstractLayoutTransform:
AbstractNoBodyTag:
AbstractNoBodyTagHelper:
AbstractObjectBrowserTabBean:
AbstractObjectBrowserTabTransform:
AbstractOptionsTransform:
AbstractPagebyContextMenuItemBuilder:
AbstractParameterBuilder.Parameter:
AbstractParameterBuilder.Parameters:
AbstractPersistableAppComponent:
AbstractPersistableFrameElement:
AbstractPreferencesTransform:
AbstractPromptExpressionTransform:
AbstractPromptObjectTransform:
AbstractReportCellComposedContext MenuItemBuilder.AbstractComposed ContextMenuBuilder:
AbstractReportCellComposedContext MenuItemBuilder.ComposedContext MenuBuilder:
AbstractReportCellComposedContext MenuItemBuilder.InvalidComposed ContextMenuBuilder:
AbstractReportCellComposedContextMenu ItemBuilder:
AbstractReportCellContextMenuItemBuilder:
AbstractReportCellInsertMetric:
AbstractReportCellMove:
AbstractReportCellParentMenu:
"AbstractReportCellSort:
AbstractReportDataTransform:
AbstractReportGridCellHeader:
AbstractReportGridCellTitle. TitlePivotButtonInfo:
AbstractReportGridCellTitle:
AbstractReportGridDisplayCell.Context:
AbstractReportGridDisplayCell. PivotButtonInfo:
AbstractReportGridDisplayCell:
AbstractReportGridTransform:
AbstractReportTransform:
AbstractRWTransform:
AbstractScheduleListTransform:
AbstractServerPropertiesTransform:
AbstractShortcut:
"AbstractShortcutList:
AbstractSubscriptionEditTransform:
AbstractSubscriptionFolderBeanTransform:
AbstractSubscriptionListTransform:
AbstractSubscriptionsEditorTransform:
AbstractTabTransform:
AbstractTransform:
AbstractViewBeanParserBean:
AbstractWebBeanTransform:
AbstractWebComponent:
AbstractWebFeatures:
ActualParameter:
ActualParameters:
AddonInfo:
AddonInfoList:
AddressDeleteTransform:
AddressListBean:
AddressListFileTransform:
AddressListPrintTransform:
AddressListTransform:
AdminBean:
AdminBeanEventHandler:
AdminBeanFactory:
AdminController:
AdminServer:
AdminServersHelper:
AdminServersList:
AdminServlet:
AdvancedSearchLinkTag:
AdvancedSearchLinkTagHelper:
AggregatedAppEventHandler:
AggregatedEventHandler:
AggregatedFolderEventHandler:
AggregatedWebFeatures:
AnchorTag:
AnchorTagImpl:
AppAddOns:
AppBeanError:
AppBeanFactory:
AppComponent:
AppContext:
AppController:
AppEventHandlerFactory:
AppGlobalContext:
AppLayoutParser:
AppLayoutSourceFile:
ArgumentTag:
AttributeFormsEditorBean:
AttributeFormsEditorTransform:
AttributeFormsElement:
AttributeFormsQualificationTransform:
AttrTag:
BaseTag:
BaseTagHelper:
BeanContext:
BeanContextImpl:
BeanDefn:
BeanDefns:
BeanHelper:
BeanValueTag:
BeanValueTagHelper:
BrowserSettingDefinition:
BrowserSettings:B
Cart:
CartAttFormsImpl:i
CartElement:
CartElementAttribute:
CartElementFolder:
CartElementPicker:
CartElementPickerImpl:
CartElementRWDataField:
CartElements:
CartElementTU:
CartElementWebAttributeForm:
CartElementWebElement:
CartElementWebObjectInfo:
CartExpression:
CartExpressionImpl:
CartFontStyles:
CartImpl:
CartObjectBrowser:
CaseInsensitiveRequestKeys:
Categories:
Category:
CellTag:
ChangePasswordBean:g
ChangePasswordGeneralTransform:
ChangePasswordPreferencesTransform:
ClientLog:
ClientSideDescriptorTag:
ClientSideDescriptorTagHelper:
ClientSideHelper:
ColorPickerEditorBean:
ColorPickerEditorTransform:
"ComponentInitializationContext:
"ComponentInitializationContextImpl:
ConfigurationElement:
ConfigurationElementList:
ConnectedServersTransform:
ConnectionValueTag:
ConnectionValueTagHelper:
ContainerServices:
ContainerStringCollection:
ContextInfo:
ContextMenu.DefaultContextMenuItem Builder:
ContextMenu:
ContextMenuItem:
ContextMenuItemBuilder:
ContextMenuManager:
ContextMenusTag:
ContextMenusTagHelper:
ControllerHelper:
ControlToolbarBean:
ControlToolbarBeanImpl:
ControlToolbarTransform:
Counter:
CounterData:
CounterDataList:
CPULicenseDetails:

CreditsBean:
DataSource:
DataSourceEnumerator:
DefaultHiddenInputBuilderImpl:
DefaultStyleRequestContext:
DefaultSubscriptionWidget:
DefaultURIBuilderImpl:
DefaultViewBeanFeaturesImpl:
DiagnosticsBean:
DiagnosticsBeanEventHandler:
DiagnosticsConfiguration:
DiagnosticsDispatcher:
DiagnosticsLogger:
DiagnosticsLoggers:
DiagnosticsPropertiesTransform:
DiagnosticsStatisticsTransform:
DiagnosticsViewerTransform:
DiagnosticsViewerTransformHelper:
DirectExportBean:
DirectExportTransform:
DisconnectedContainerServices:
DisconnectedContainerServicesImpl:
DisplayBeanTag:
DisplayBeanTagHelper:
DisplayCookiesTag:
DisplayCookiesTagHelper:
DisplayCookiesTagHelper:
DisplayGuiComponentTag:
DisplayGuiComponentTagHelper:
DisplayUnitsVisitor:
DocumentBean:
DocumentDetailsTransform:
DocumentDetailsXHTMLTransform:
DocumentExportTransform:
DocumentFastExportTransform:
DocumentFrameBean:
DocumentFrameElement:
DocumentImageServlet:
DocumentSetFlagsAddOn:
DocumentSimpleTransform:
DragAndDropItem:
DrillEditorBean:
DynamicMenus:
DynamicTag:
DynTableCellTag:
DynTableCellTagHelper:
DynTableTag
DynTableTagHelper:
EditableBean:
EditableObject:d
Editor:
EditorImpl.Button:
EditorImpl:
ElementPickerBean:
ElementPickerBeanImpl:
ElementPickerTransform:
ElseTag:
EmptyNamespaceEncoderImpl:
EmptyNumericStringPreferenceDefinition:
EnumAddressListBeanEvents:
EnumAdminBeanEvents:
EnumAppBeanViewModes:
EnumAppComponentEvents:
EnumAppWebFeatures:
EnumAttributeFormsEditorEvents:
EnumBeanErrorCodes:
EnumBeanOutputFormat:
EnumBeanPersistMode:
EnumCartProperties:
EnumChangePasswordEvents:
EnumColorPickerBeanEvents:
EnumContextMenuItems:
EnumCSSClasses:
EnumCustomWizardStepBeanEvents:
EnumDefaultServerProperties:
EnumDiagnosticsBeanEvents:
EnumDocumentBeanEvents:
EnumDocumentFrameEvents:
EnumDragAndDropProperties:
EnumDrillEditorEvents:
EnumDrillFilterEditorEvents:
EnumEditableObjectBeanEvents:
EnumEditorDisplayFlags:
EnumEditorElements:
EnumEditorSectionTypes:
EnumEventElement:
EnumEventHandlerTypes:
EnumExcelVersion:
EnumExecutionMode:
EnumExecutionScope:
EnumExportFormats:
EnumExportPlaintextDelimiters:
EnumExportSection:
EnumExpressionBeanEvent
EnumExpressionCartTransformProperties:
EnumExpressionEditMode:
EnumFilterEditorErrorCo
EnumFilterElementEvents:WebEvent
ArgumentAutoApply:int
EnumFilterElementEvents:
WebEventToggleAutoApply:int
EnumFilterTypes:
EnumFolderBeanEvents:
EnumFormatBorderFlags:
EnumFormatOutputTypes:
EnumFormatTabManagerEvents:
EnumFormatToolbarEvents:
EnumFormulaBarEvents:
EnumFrameEditorEvents:
EnumFrameEvents:
EnumGenericEvents:
EnumGraphFormatAxisParameters:
EnumGraphFormatEditorEvents:
EnumGraphFormatEditorTabs:
EnumGraphFormatTextParameters:
EnumGraphFormatToolbarEvents:
EnumGridAutoStylesEvents:
EnumGridFormatEditorEvents:
EnumGridFormatEditorTabs:
EnumGridFormatEvents:
EnumGridFormatGenericValues:
EnumGridFormatNumberCategory:
EnumGridFormatToolbarEvents:
EnumGridFormatUnit:
EnumGridHorizontalAlignmentType:
EnumGridNegativeNumbersFormat:
EnumGridNumberCurrencyPosition:
EnumGridVerticalAlignmentType:
EnumGroupBeanEvents:
EnumGroupbyPropertiesEditorEvents:
EnumGuiElements:
EnumHTMLAttributes:
EnumHTMLTags:
EnumInboxBeanEvents:
EnumIncrementalFetchProperties:
EnumLogDestinationType:
EnumMenuImages:
EnumObjectBeanEvents:
EnumObjectBrowserBeanEvents:
EnumObjectBrowserContexts:
EnumObjectBrowserManagerBeanEvents:
EnumObjectBrowserTabs:
EnumObjectBrowserTargets:
EnumObjectManipulationDialogEvents:
EnumOMDCreateFolderEvents:
EnumPageByBeanEvents:
EnumPageEvents:
EnumPageInfoDirection:
EnumParameterType:
EnumPreferenceLevels:
EnumPreferencesEvents:
EnumPrintEvents:W
EnumProjectBrowserEvents:
EnumPromptAnswerFormat:
EnumPromptAnswerOptions:
EnumPromptLevelFlags:
EnumPromptsBeanEvents:
EnumPromptsBeanTypes:
EnumPromptsSourceTypes:
EnumReportBeanEvents:
EnumReportDesignMode:
EnumReportFrameEvents:
EnumReportGridDisplayCellTypes:
EnumReportPageType:
EnumReportThresholdSymbols:
EnumReportViewModes:
EnumRequestStatus:
EnumRWBeanEvents:
EnumRWExecutionModes:
EnumRWFrameEvents:
EnumRWGroupByLevels:
EnumRWGroupByTypes:
EnumRWProjectBrowserBeanEvents:
EnumRWResultFilterModes:
EnumRWUnitTypes:
EnumSaveAsEvents:
EnumScheduleBeanEvents:
EnumSearchWildcards:
EnumSecurityFilterBeanEvents:
EnumServerState
EnumServerTypes
EnumServiceMode
EnumServletEvents:
EnumSortBy:
EnumSortEditorEvents:
EnumSubscriptionBeanEvents:
EnumSubscriptionEditorEvents:
EnumSubscriptionFolderBeanEvents:
EnumSubtotalsEditorEvents:
EnumSysBeanNames:
EnumSysBeanTypes:
EnumTabManagerBeanEvents:
EnumTemplateFilterExecEvents:
EnumToolbarBeanEvents:
EnumToolbarSetBeanEvents:
EnumUserBeanEvents:
EnumUserEntityBeanEvents:
EnumViewBeanEvents:
EnumWebAppDebugFlags:
EnumWebAppErrorCodes:
EnumWebAttributeFormNamesDisplay:
EnumWebBrowserSettings:
EnumWebConfigBeanType:
EnumWebControllers:
EnumWebDependentObjectType:
EnumWebDimensionDisplayStyle:
EnumWebDisplayedForms:
EnumWebDocumentViewMode:
EnumWebDrillProperties:
EnumWebElementSourceType:
EnumWebFeatures:
EnumWebFolderDisplayStyle:
EnumWebFormatType:
EnumWebFunctionType:
EnumWebHeaderType:
EnumWebLimitSummaryFlags:
EnumWebMessageSortByField:
EnumWebMonitorStatistics:
EnumWebMRPFunction:

EnumWebNamedUsers:
EnumWebObjectsFeatures:
EnumWebObjectSort:
EnumWebPages:
EnumWebParameters:
EnumWebPreferences:
EnumWebPromptType:
EnumWebPropertySource:
EnumWebReportExecutionModes:
EnumWebReportExportModes:
EnumWebReportViewMode:
EnumWebRowValueType:
EnumWebScheduleSort:
EnumWebSessionType:
EnumWebStateLevel:
EnumWebStatementType:
EnumWebSubscriptionContentCompression Status:
EnumWebSubscriptionContentFormatModes:
EnumWebSubscriptionContentFormatTypes:
EnumWebSubscriptionContentTypes
EnumWebSubscriptionDeliveryMode:
EnumWebSubscriptionDeviceSubtype:
EnumWebSubscriptionObjectTypes:
EnumWebToggleFunction:
EnumWebTransformableType:
EnumWebWorkingSetSortByField:
EnumWizardBeanEvents:
EnumWizardStepBeanEvents:
EnumXmlStatePhase:
ErrorInfo:
ErrorInfoList:
ErrorValueTag:
ErrorValueTagHelper:
EventArgumentTag:
EventFlagList:
EventFlagListSet:
EventHandlerInfo:
EventHandlerList:
EventHelper:
EventManager:
EventMap:
EventMapList:
EventTag:
EventTagHelper:
ExportBean:
ExportBeanHelper:
ExportFormat:
ExportFormatList:
ExportOptionsEditorBean:
ExportOptionsEditorTransform:
ExportReportAddOn:
ExportSaveReportPropertiesAddOn:
ExportSetContentTypeTag:
ExportSetContentTypeTagHelper:
ExportTransform:
ExpressionBean:
ExpressionFilterTransform:
ExpressionGenericTransform.Context:
ExpressionGenericTransform:
ExpressionHelper:
ExpressionMessages:
ExpressionObject:
ExpressionParts:
ExpressionPartsImpl:
ExpressionPlainTextTransform:
ExternalPromptsSource:
ExternalSecurity:
FastExportAddOn:
FastReportPDFExportAddOn:
FeatureArray:
FeaturesHelper:
FileContentsObjectList:
FileContentsObjectWrapper:
FileLogDestination:
FileUtils:
FilterAnd:
FilterAndNot:
FilterEditorBean:
FilterElementBean:
FilterElementRWTransform:
FilterElementTransform:
FilterOr:
FilterOrNot:
FlagInfo:
FolderAddDataSetTransform:
FolderBean:
FolderBulletReportsTransform:
FolderBulletRWTransform:
FolderBulletTransform:
FolderCopy:
FolderCreateDocumentTransform:
FolderCreateReportTransform:
FolderCreateReportXHTMLTransform:
FolderCreateShortcut:
FolderDelete:
FolderDeleteObjectTransform:
FolderDetailsTransform:
FolderDisplayProperties:
FolderIconRadioButtonTransform:
FolderIconTransform:
FolderLastVisitedAddOn:
FolderLink:
FolderListTransform:
FolderMapping:
FolderMappingList:
FolderMove:
FolderObjectBrowserTransform:
FolderOMDBrowserTransform:
FolderProjectBrowserAddOn:
FolderProjectBrowserAddOnBase:
FolderProjectBrowserTransform:
FolderRadioButtonTransform:
FolderRename:
FolderReportProjectBrowserAddOn:
FolderSaveAsBrowserTransform:
FolderSaveAsTransform:
FolderSaveAsXHTMLTransform:
FolderSearchResultsTransform:
FolderSearchResultsXHTMLTransform:
FolderSetFlagsAddOn:
FolderSmallIconTransform:
FolderSummaryCreateDocumentTransform:
FolderSummaryCreateReportTransform:\
FolderSummaryTransform:
FolderViewTransform:
FormalParameter:
"FormalParameterAnnotation:
FormalParameterImpl:
FormalParameters:
FormatColorLinesTabBean:
FormatEditorTabManagerTransform:
FormatNumberTabBean:
FormatTabAlignmentTransform:
FormatTabBean:
FormatTabColorLinesTransform:
FormatTabFontTransform:
FormatTabManagerBean:
FormatTabNumberTransform:
FormatToolbarBean:
FormatToolbarTransform:
FormEventTag:
FormEventTagHelper:
FormulaBarBean:
FormulaBarTransform:
FrameBean:
FrameElement:
FrameGenericTransform:
GenericBrowserSettings:
GenericCookie:
GenericEventHandler:
GenericMultipleEvents:
GenericRequestKeys:
GenericWebEvent:
GGCalendar:
GGCalendarImpl:
GlobClassMethodTracingFilter:
GraphAxisInformation:
GraphBean:
GraphFormatEditorBean:
GraphFormatHelper:
GraphFormatToolbarBean:
GraphImageTransform:
GraphSeriesInformation:
GraphTitleInformation:
GridAutoStylesBean:
GridAutoStylesHelper:
GridAutoStylesTransform:
GridFormatBean:
GridFormatEditorBean:
GridFormatHelper:
GridFormatToolbarBean:
GroupbyPropertiesEditorBean:
GroupbyPropertiesEditorTransform:
GuiComponent:
GuiElement:
GuiElementFactory:
HighlightedObject:
HighlightedObjects:
HTMLAttributesJsGenerator:
HTMLHelper:
HttpRequestKeys:
IfBeanValueTag:
IfBeanValueTagHelper:
IfConnectionValueTag:
IfConnectionValueTagHelper:
IfDisplayMoreTag:
IfDisplayMoreTagHelper:
IfErrorValueTag:
IfErrorValueTagHelper:
IfFeatureTag:
IfFeatureTagHelper:
IfPlainTextExportTag:
IfPlainTextExportTagHelper:
IfShowAdminPageTag:
IfShowAdminPageTagHelper:
IfTag:
IfTemplateValueTag:
IfTemplateValueTagHelper:
IfUseIframeTag:
IfUseIframeTagHelper:
ImageTag:
InboxBean:
InboxClearTransform:
InboxDetailsTransform:
InboxListTransform:
InboxListXHTMLTransform:
InboxToolbarTransform:
IncrementalFetch:
IncrementalFetchImpl:
InputTag:
Instance:
InstanceStatus:
IntelligenceServerInstance:
IntelligenceServerInstanceStatus:
JavaScriptHelper:
JavaScriptInfo:
JavaScriptInfoList:
JavaScriptTag:

JavaScriptTagHelper:
JsonGenerator:
Layout:
LayoutContext:
LayoutParser.
LayoutParserDefn:
LayoutParserDefns:
Layouts:
LayoutSource:
LayoutSourceDefn:
LayoutSourceDefns:
LayoutSourceFile:
LayoutSourceString:
LayoutTag:
LayoutTransform:
LicensedUsers:
LicenseSource:
LicensingInfoTag:
LicensingInfoTagHelper:
ListTag:
Log:
LogCategories:
LogCategory:
LogCounter:
LogDestination:
LogDestinations:
LogFormatter:
LoggingTag:
LoggingTagHelper:
LoginBean:
LoginContinueTransform:
LoginForm:
LoginTransform:
LoginXHTMLTransform:
LogoutBean:
LogoutTransform:
MachineCPUInfo:
MacroHelper:
MainServlet:
MarkupOutput:
Menu:
MenuImpl:
MessageInfo:
Messages:
MessagesManager:
MetaContentType:
MetaContentTypeTag:
MetaContentTypeTagHelper:
MetadataPromptsSource:
MetricQualLevelEditorBean:
MetricQualLevelEditorTransform:
MSTRLayoutParser.ParserContentHandler:
MSTRLayoutParser.ParserEntityResolver:
MSTRLayoutParser.ParserErrorHandler:
MSTRLayoutParser:
MSTRLoginForm:
MstrPageTagHelper:
MSTRWebController:
MSTRWebRequest:
MultipartRequest.File:
MultipartRequest:
NamedUserLicense:
NamespaceEncoder:
NextTag:
ObjBrowserFolderLinks:
ObjectBean:
ObjectBrowserBean:
ObjectBrowserManagerBean:
ObjectBrowserManagerTransform:
ObjectBrowserTabBean:
ObjectBrowserTabOMDTransform:
ObjectBrowserTabProjectBrowserTransform:
ObjectBrowserTabReportAllObjectsTransform:
ObjectBrowserTabReportWorkingSetTransform:
ObjectBrowserTabRWWorkingSetTransform:
ObjectInfoTransform:
ObjectInfoWidget:
ObjectManipulationDialogBean:
ObjectManipulationDialogTransform:
ObjectPathTag:
ObjectPathTagHelper:
ObjectPathTransform:
ObjectPathXHTMLTransform:
ObjectReportPathTransform:
ObjectReportXHTMLPathTransform:
ObjectSiblingsTransform:
OfficeTransform:
OldLinksController:
OldLinksRequest:
OldLinksServlet:
OMDCreateFolderBean:
OMDCreateFolderTransform:
OptionsAddOn:
OptionsBean:
OptionsBeanImpl:
OptionsHelper:
PageByBean:
PageByDelete:
PageByElement:
PageByElementList:
PageByPivot:
PageByProperties:
PageByRemove:
PageByRename:
PageBySort:
PageByTransform.Context:
PageByTransform:
PageByUnit:
PageComponent:
PageInfo:
PageManager:
PageStateTag:
PageStateTagHelper:
PageSubscriptionsEdit:
Panel_CloseInfoTag:
Panel_ContentTag:
Panel_OpenInfo:
Panel_PanelTag:
Panel_TitleTag:
PanelTagHelper:
PaperSizes:
PaperSizesList:
ParameterBuilder:
PDFBean:
PDFSaveReportPropertiesAddOn:
PDFTransform:
PerformanceLoggingConfiguration:
PerformanceMonitorBean:
PerformanceMonitorBeanManager:
PerformanceMonitorInstance:
PerformanceMonitorManager:
PerformanceStringTag:
PerformanceStringTagHelper:
PerformanceTimerTag:
PerformanceTimerTagHelper:
PostedFile:
PreferenceBrowserSettings:
PreferenceDefinition:
PreferenceDefinitionBase:
PreferenceDefinitionImpl:
PreferenceDefinitionMinMax:
PreferenceLevel:
PreferenceLevels:
Preferences:
PreferencesBean.GroupInfo:
PreferencesBean:
PreferencesBeanImpl.GroupInfoImpl:
PreferencesBeanImpl:
PreferencesDocumentTransform:
PreferencesDrillTransform:
PreferencesException:
PreferencesExportTransform:
PreferencesFolderTransform:
PreferencesGeneralTransform:
PreferencesGraphTransform:
PreferencesGridProjectLevelTransform:
PreferencesGridTransform:
PreferencesInboxTransform:
PreferencesLogoutTransform:
PreferencesMgr:
PreferencesNCTransform:
PreferencesOfficeTransform:
PreferencesPDFTransform:
PreferencesPrintHeaderFooterTransform:
PreferencesPrintTransform:
PreferencesProjectTransform:
PreferencesPromptsTransform:
PreferencesSecurityTransform:
PreferencesUserGroupsTransform:
PreferenceValidationForMargins:
PreferenceValidationForPrintRowsColsPerPage:
PreferenceValueTag:
PreferenceValueTagHelper:
PrettyPrintTransform:
PrintBean:
PrintBeanHelper:
PrintSaveReportPropertiesAddOn:
PrintTransform:
ProjectBrowser:
ProjectBrowserEventHandler:
ProjectBrowserImpl:
ProjectCacheBase:
ProjectInfo:
ProjectInformation:
ProjectsBean:
ProjectStatusBean:
ProjectStatusTransform:
ProjectsTransform:
PromptCartTransform:
PromptCartXHTMLTransform:
PromptCheckboxTransform:
PromptConstantTransform:
PromptDetailsTransform:
PromptExpressionCartTransform:
PromptExpressionCartWithPickerTransform:
PromptExpressionCartXHTMLTransform:
PromptExpressionListTransform:
PromptExpressionNonCartTransform:
PromptExpressionTextboxTransform:
PromptExpressionTransform:
PromptHierarchicalCartTransform:
PromptHierarchicalCartXHTMLTransform:
PromptHierarchicalTreeTransform:
PromptHierarchicalTreeXHTMLTransform:
PromptListboxTransform:
PromptObject:
PromptObjectBrowsingTransform:
PromptObjectBrowsingXHTMLTransform:
PromptRadioTransform:
PromptsBean:
PromptsClassicTransform:
PromptsCreateReportTransform:
PromptsIFrameTransform:
PromptsSource:
PromptsSubscriptionTransform:
PromptsSummaryTransform:
PromptXSLTransform:
Property:
PropertySet:

PropertySetHelper:
PropertySets:
PropertyValue:
QuickLinks:
QuickLinksImpl:
QuickSearchTag:
QuickSearchTagHelper:
RenderTag:
ReportBean:
ReportCellAttributeForms.AbstractAttributeFormsContextMenuBuilder:
ReportCellAttributeForms:
ReportCellDrill.AbstractDrillContextMenuBuilder:
ReportCellDrill:
ReportCellFilterOn:
ReportCellFormat:
ReportCellInsertMetric:
ReportCellInsertPercentToTotalMetric:
ReportCellMove:
ReportCellMoveDown:
ReportCellMoveLeft:
ReportCellMoveRight:
ReportCellMoveToColumns:
ReportCellMoveToPageBy:
ReportCellMoveToRows:
ReportCellMoveUp:
ReportCellPercentForEachMetric.AbstractInsertMetricContextMenuBuilder:
ReportCellPercentForEachMetric:
ReportCellPercentGrandTotalMetric:
ReportCellPercentOverColumnsMetric:
ReportCellPercentOverPagesMetric:
ReportCellPercentOverRowsMetric:
ReportCellRankMetric:
ReportCellRemoveFromGrid:
ReportCellRemoveFromReport:
ReportCellRename:
ReportCellSort:
ReportCellSortAscending:
ReportCellSortDescending:
ReportCellTransformationMetric.AbstractInsertMetricContextMenuBuilder:
ReportCellTransformationMetric:
ReportDataTransform:
ReportDesignModeTransformImpl:
ReportDetailsSetFlagsAddOn:
ReportDetailsTransform:
ReportDetailsXHTMLTransform:
ReportExportExcelFormattingTransform:
ReportExportHTMLTransform:
ReportExportPlainTextTransform:
ReportFrameBean:
ReportFrameBeanImpl:
ReportFrameClassicTransform:
ReportFrameElement:
ReportGraphImageTransform:
ReportGraphTransformImpl:
ReportGridCellColHeader:
ReportGridCellColHeaderImpl.ColHeaderPivotButtonInfo:
ReportGridCellColHeaderImpl:
ReportGridCellColTitleImpl:
ReportGridCellHeader:
ReportGridCellImpl:
ReportGridCellMetricValue:
ReportGridCellMetricValueImpl:
ReportGridCellRowHeader:
ReportGridCellRowHeaderImpl:
ReportGridCellRowTitle:
ReportGridCellRowTitleImpl:
ReportGridCellsFactory:
ReportGridCellTitle:
ReportGridDisplayCell.Context:
ReportGridDisplayCell:
ReportGridDisplayCellColHeader:
ReportGridDisplayCellColTitle:
ReportGridDisplayCellImpl:
ReportGridDisplayCellMetricValue:
ReportGridDisplayCellRowHeader:
ReportGridDisplayCellRowTitle:
ReportGridGraphTransformImpl:
ReportIncrementalFetchTransform:
ReportLastUpdateTransform:
ReportOutlineModeTransformImpl:
ReportPageByInfoTransform:
ReportPageByTransform:
ReportPageInfo:
ReportQuickLinksTransform:
ReportSavePropertiesTransform:
ReportSetFlagsAddOn:
ReportToolbarTransform:
ReportTransformHelper:
ReportWSDelete:
RepositorySettings:
RepositorySource:
RequestKeys:
RequestPersistable:
RequestState:
RequestStringTag:
RequestStringTagHelper:
ResizeEditorBean:
ResizeEditorTransform:
Resource:
ResourceCache:
ResourceCacheHint:
ResourceCacheManager:
ResultSetBean:
ReturnToPathTag:
ReturnToPathTagHelper:
ReturnToTag:
ReturnToTagHelper:
RowTag:
RWBean:
RWClipboard:
RWData:
RWDataSet:
RWDataSets:
RWDefinition:
RWDesignModeAddOn:
RWDetailsTransform:
RWDFilterAddOn:
RWDWizardAddOn:
RWExportAddOn:
RWExportSettings:
RWExportTransform:
RWFieldDef:
RWFormatProperty:
RWFrameBean:
RWGridGraphDef:
RWGridGraphObject:
RWGroupBy:
RWGroupByElement:
RWGroupBys:
RWGroupByUnit:
RWImageDef:
RWInstance:
RWLastUpdateTransform:
RWLayoutTransform:
RWManipulation:
RWMessage:
RWObject:
RWPDFTransform:
RWProjectBrowserTabBean:
RWSavePropertiesTransform:
RWSection:
RWSectionDef:
RWShapeDef:
RWSource:
RWTextDef:
RWThreshold:
RWUnit:
RWUnitDef:
RWUserMetric:
RWVisitor:
SampleTransform:
SaveAsBean:
SaveAsFolderFlagsAddOn:
SaveAsReportTransform:
SaveAsRWTransform:
SaveAsTransform:
ScheduleBean:
ScheduleEditTransform:
ScheduleOverwriteTransform:
SchedulesBulletTransform:
SchedulesFilteredListTransform:
SchedulesFilteredListXHTMLTransform:
SchedulesIconTransform:
SchedulesIconXHTMLTransform:
SchedulesListTransform:
SchedulesListXHTMLTransform:
ScheduleWidget:
ScriptPageStateTag:
ScriptPageStateTagHelper:
Scrollable:
SearchBean:
SearchSimpleTransform:
SectionInfo:
SectionInfoList:
SecurityFilterBean:
SecurityFilterEventHandler:
SecurityTransform:
SelectTag:
ServerCacheBase:
ServerControlException:
ServerControlSource:
ServerInstance:
ServerInstances:
ServerMachine:
ServerPropertiesTransform:
ServerVersionInfo:
ServiceConfiguration:
ServletContainerServices:
ServletEvent:
ServletWebComponent:
SessionCacheBase:
SessionManager:
Shortcut:
ShortcutArgument:
ShortcutBean:
ShortcutButton:
ShortcutCaption:
ShortcutColorPickerPane:
ShortcutCombo:
ShortcutDefault:
ShortcutElement:
ShortcutElementTag:
ShortcutEvent:
ShortcutGridInfo:
ShortcutGridLevels:
ShortcutHelper:
ShortcutInput:
ShortcutList:
ShortcutListElement:
ShortcutListPicker:
ShortcutListSet.EventActionMapping:
ShortcutListSet:
ShortcutListStyles:
ShortcutOptionsTag:

ShortcutPropertyTag:
ShortcutPulldown:
ShortcutSeparatorTag:
ShortcutTagHelper:
SimpleList:
SiteDesigner:
SortDefinition:
SortEditorBean:
SortEditorTransform:
SourceLookup:
StaticSession:
StaticSessionList:
StatsHandler:
StatsHandlerSupport
StringRequestKeys:
StringTag:
StringTagHelper:
Style:
StyleCatalog:
StyleCatalogFactory:
StyleRequestContext:
Styles:
SubscriptionBean:
SubscriptionConfirmSendNowTransform:
SubscriptionConfirmTransform:
SubscriptionEditFileTransform:
SubscriptionEditorBean:
SubscriptionEditPrintTransform:
SubscriptionEditTransform:
SubscriptionFileWidget:
SubscriptionFileWidgetImpl:
SubscriptionFolderBean:
SubscriptionPrintWidget:
SubscriptionPrintWidgetImpl:
SubscriptionsBulletTransform:
SubscriptionsEditorNCFileTransform:
SubscriptionsEditorNCPrintTransform:
SubscriptionsEditorNCSendNowTransform:
SubscriptionsEditorNCTransform:
SubscriptionsEditorTransform:
SubscriptionSendNowTransform:
SubscriptionsFilteredListTransform:
SubscriptionsFilteredListXHTMLTransform:
SubscriptionsIconTransform:
SubscriptionsIconXHTMLTransform:
SubscriptionsListTransform:
SubscriptionsListXHTMLTransform:
SubscriptionWidget:
SubtotalDefinition:
SubtotalsEditorBean:
SubtotalsEditorTransform:
SysDefaultPrefType:
SysPrefType:
SystemPickerList:
TabBean:
TabBeanTransform:
TabControlHelper:
TabDocumentOtherTransform:
TabGraphAxesTransform:
TabGraphGeneralTransform:
TabGraphTitlesTransform:
TableTag:
TabManagerBean:
TabManagerTransform:
TabPageMarginTransform:
TabPageOutputTransform:
TabPageSetupTransform:
TabPropertiesGeneralTransform:
TabPropertiesGridTransform:
TabPropertiesLayoutTransform:
TabPropertiesLineTransform:
TabPropertiesPictureTransform:
Tag:
TagsFactory:
TemplateElement:
TemplateFilterExecBean:
TemplateFilterExecTransform:
TemplateFilterExecXHTMLTransform:
TemplateInfo:
TemplateInfoList:
TemplateParser:
TestBodyContentTag:
ThenTag:
ThreadSessionMap:
Toolbar:
ToolbarBean:
ToolbarInfoList:
ToolbarSetBean:
ToolbarSetTransform:
ToolbarTransform:
Transform:
Transformable:
TransformContext:
TransformDefn:
TransformInstance:
Transforms:
TreeBox:
TreeBoxImpl:
TreeCartExpressionImpl:
TreeNode:
UnconnectedServersTransform:
UpdateManagerHelper:
UpdateManagerTag:
UpdateManagerTagHelper:
UrlEventTagHelper:
UserEntitiesBean:
UserGroupBean:
UserLicenseAudit:
UserSearchBean:
ValueTag:
ValueTagHelper:
ViewBean:
ViewBeanParserBean:
ViewBeanTransformHelper:
WaitTag:
WaitTagHelper:
WebAccessControlEntry:
WebAccessControlList:
WebAppBeanErrorImpl:
WebAppFactory:
WebAppRuntimeException:
WebAppSessionManager:
WebAttribute:
WebAttributeForm:
WebAttributeForms:
WebAttributeHeader:
WebAttributeList:
WebAxis:
WebBean:
WebBeanError:
WebBeanFactory:
WebBeanInfo:
WebBeanInfoList:
WebBeanProperty:
WebBeanPropertyList:
WebBeanStyle:
WebBeanStyleList:
WebBeanUtils.SimpleRequestKeys:
WebBeanUtils:
WebCacheStatement:
WebChannel:
WebChannels:
WebCluster:
WebClusterAdmin:
WebClusterNode:
WebClusters:
WebColumn:
WebColumns:
WebComponent:
WebComponentFactory:
WebConstantNode:
WebConstantPrompt:
WebContentStatement:
WebCSSHelper:
WebDatamart:
WebDatamartTable:
WebDefaultDisplaySettings:
WebDefaultSort:
WebDimension:
WebDimensionAttribute:
WebDimensionHeader:
WebDimty:
WebDimtyPrompt:
WebDimtyUnit:
WebDirectedAttribute:
WebDisplayHelper:
WebDisplayUnit:
WebDisplayUnitEntry:
WebDisplayUnits:
WebDocReportInstances:
WebDocumentMessage:
WebDocumentSource:
WebDrillAction:
WebDrillActions:
WebDrillElements:
WebDrillInstance:
WebDrillMap:
WebDrillPath:
WebEditablePrivileges:
WebElement:
WebElements:
WebElementsObjectNode:
WebElementSource:
WebElementsPrompt:
WebEvent.Argument:
WebEvent:
WebEventHandler:
WebEvents:
WebEventTags:
WebExportHelper:
WebExpression:
WebExpressionHelper:
WebExpressionPrompt:
WebFeatures:
WebFilter:
WebFilterEditorException:
WebFolder:
WebFormatContainer:
WebFormShortcutNode:
WebFormSort:
WebFunction:
WebGraph:
WebGraphCoordinatesDrillAction:
WebGraphProperties:
WebGraphProperty:
WebGridData:
WebGridHeaders:
WebGridRows:
WebGridTitles:
WebGridWidths:
WebGuiComponent:
WebGuiComponentList:
WebHeader:
WebHeaders:
WebInboxSource:
WebIServerSession:
WebIServerSessionList:
WebIServerSessionMgr:
WebIServerSessionWrapper:

WebLDAPLoginInfo:
WebLDAPSettings:
WebLicenseDetails:
WebLink:
WebLinkConfigurationCache:
WebLinkItem:
WebMDSecurityFilter:
WebMessage:
WebMessages:
WebMessageStatusList:
WebMetricHeader:
WebMetricHierarchicalSort:
WebMetricSort:
WebMonitor:
WebMonitorField:
WebMonitorProperty:
WebNewFormsDrillAction:
WebNewObjectDrillAction:
WebNode:
WebNTLoginInfo:
WebObjectInfo:
WebObjectSecurity:
WebObjectsFactory:
WebObjectSource:
WebObjectsPrompt:
WebOperatorNode:
WebOptimizedDrillPathSettings:
WebPDFHelper:
WebPDFSettings:
WebPercentToTotalMetric:
WebPerformanceMonitor:
WebPrivilegeCategories:
WebPrivilegeCategory:
WebPrivilegeEntry:
WebPrivilegeOrigin:
WebPrivileges:
WebProjectInstance:
WebProjectInstances:
WebProjectReference:
WebProjectReferences:
WebProjectSetting:
WebProjectSettings:
WebProjectSource:
WebProjectStatus:
WebPrompt:
WebPrompts:
WebProperties:
WebProperty:
WebPropertyGroup:
WebPropertySet:
WebRankMetric:
WebRDBMSLoginInfo:
WebRelationship:
WebRelationshipNode:
WebRemoveMetricDrillAction:
WebReportData:
WebReportExcelExportSettings:
WebReportExportSettings:
WebReportGrid:
WebReportInstance:
WebReportManipulation:
WebReportMessage:
WebReportPDFExportSettings:
WebReportPlainTextExportSettings:
WebReportSource:
WebReportStatement:
WebReportValidationException:
WebRequest:
WebResultSetInstance:
WebResultSetManipulation:
WebResultSettings:
WebResultWindow:
WebRow:
WebRowValue:
WebSchedule:
WebSchedules:
WebScheduleSource:
WebSearch:
WebSecurityRole:
WebServerDef:
WebServerSetting:
WebServerSettings:
WebSessionInfo:
WebSessionInfoList:
WebShortcut:
WebShortcutNode:
WebSort:
WebSorts:
WebStandardLoginInfo:
WebStatement:
WebSubscription:
WebSubscriptionAddress:
WebSubscriptionComponent:
WebSubscriptionContent:
WebSubscriptionContentDocument:
WebSubscriptionContentFormat:
WebSubscriptionContentProperties:
WebSubscriptionContentReport:
WebSubscriptionContentReportProperties:
WebSubscriptionDeliveryModeEmailProperties:
WebSubscriptionDeliveryModeEmail PropertiesImpl:
WebSubscriptionDeliveryModeFileProperties:
WebSubscriptionDeliveryModePrintProperties:
WebSubscriptionDeliveryModeProperties:
WebSubscriptionDevice:
WebSubscriptionHelper:
WebSubscriptionLocation:
WebSubscriptionLocationAttributes:
WebSubscriptionNotification:
WebSubscriptionProperties:
WebSubscriptionsSource:
WebSubscriptionTrigger:
WebSubTitle:
WebSubTitles:
WebTable:
WebTemplate:
WebTemplateAttribute:
WebTemplateConsolidation:
WebTemplateCustomGroup:
WebTemplateDimension:
WebTemplateFormatContainer:
WebTemplateMetric:
WebTemplateMetrics:
WebTemplateMetricSubtotal:
WebTemplateMetricSubtotals:
WebTemplateUnit:
WebTimeNode:
WebTitle:
WebTransformationMetric:
WebUser:
WebUserEntity:
WebUserGroup:
WebUserList:
WebUserSearch:
WebUserSecurityFilters:
WebUserSecurityRoles:
WebUserServicesSource:
WebValueNode:
WebViewInstance:
WebWorkingSet:
WebWrapperAttributeList:
WizardBean:
WizardStepBean:
WizardStepTransform:
WizardTransform:
XMLValidator:

MicroStrategy Office API

MOIOfficeInterface.Document.ExecuteAnd DisplayDefaultInExcel
MOIOfficeInterface.Document.ExecuteAnd DisplayDefaultInPowerPoint
MOIOfficeInterface.Document.ExecuteAnd DisplayDefaultInWord
MOIOfficeInterface.Document. ExecuteDocument
MOIOfficeInterface.Document.Initialize
MOIOfficeInterface.DocumentInstance. AnswerPrompt
MOIOfficeInterface.DocumentInstance. GetDocumentResults
MOIOfficeInterface.DocumentInstance. GetPrompts
MOIOfficeInterface.EnumMOIAggregationType
MOIOfficeInterface.EnumMOIDisplayType
MOIOfficeInterface.EnumMOIErrorStatus
MOIOfficeInterface.EnumMOIExecutionType
MOIOfficeInterface.EnumMOIFilteringType
MOIOfficeInterface.EnumMOIFunctionType
MOIOfficeInterface.EnumMOIOperatorType
MOIOfficeInterface.EnumMOIPromptType
MOIOfficeInterface.Folder.Contents
MOIOfficeInterface.Folder.FolderID
MOIOfficeInterface.Folder.FolderName
MOIOfficeInterface.Folder.GetContents
MOIOfficeInterface.Folder.Initialize
MOIOfficeInterface.Folders.Count
MOIOfficeInterface.Folders.Item
MOIOfficeInterface.HistoryItem. CancelHistoryItem
MOIOfficeInterface.HistoryItem.Description
MOIOfficeInterface.HistoryItem.ExecuteAnd DisplayDefaultInExcel
MOIOfficeInterface.HistoryItem.ExecuteAnd DisplayDefaultInWord
MOIOfficeInterface.HistoryItem.ExecuteAnd DisplayDefaultnPowerPoint
MOIOfficeInterface.HistoryItem. ExecuteAndDisplayInExcel
MOIOfficeInterface.HistoryItem. ExecuteAndDisplayIn PowerPoint
MOIOfficeInterface.HistoryItem. ExecuteAndDisplayIn Word
MOIOfficeInterface.HistoryItem.ExecuteReport
MOIOfficeInterface.HistoryItem.FinishTime
MOIOfficeInterface.HistoryItem.Folder
MOIOfficeInterface.HistoryItem. GetHistoryResults
MOIOfficeInterface.HistoryItem.HistoryItemID
MOIOfficeInterface.HistoryItem. HistoryItemName
MOIOfficeInterface.HistoryItem. HistoryItemSubType
MOIOfficeInterface.HistoryItem.HistoryItemType
MOIOfficeInterface.HistoryItem.LastUpdateTime
MOIOfficeInterface.HistoryItem.Message
MOIOfficeInterface.HistoryItem.MessageID
MOIOfficeInterface.HistoryItem.Owner
MOIOfficeInterface.HistoryItem.StartTime
MOIOfficeInterface.HistoryItems.Count
MOIOfficeInterface.HistoryItems.Item
MOIOfficeInterface.MOIOffice.Connect
MOIOfficeInterface.MOIOffice. DeleteDisplayedElement
MOIOfficeInterface.MOIOffice.DisplayReport XMLFileInExcel
MOIOfficeInterface.MOIOffice.DisplayReport XMLFileInPowerPoint
MOIOfficeInterface.MOIOffice.DisplayReport XMLFileInWord

MOIOfficeInterface.MOIOffice.DisplayReportXMLInExcel
MOIOfficeInterface.MOIOffice.DisplayReportXMLInPowerPoint
MOIOfficeInterface.MOIOffice.DisplayReportXMLInWord
MOIOfficeInterface.MOIOffice.GetErrorMessage
MOIOfficeInterface.MOIOffice.GetPersistedReportLocations
MOIOfficeInterface.MOIOffice.GetProjectSources
MOIOfficeInterface.MOIOffice.Initialize
MOIOfficeInterface.MOIOffice.InitializeDocument
MOIOfficeInterface.MOIOffice.RefreshAllReports
MOIOfficeInterface.MOIOffice.RefreshDisplayedElement
MOIOfficeInterface.MOIOffice.Uninitialize
MOIOfficeInterface.ObjectInfo.Access
MOIOfficeInterface.ObjectInfo.CreationTime
MOIOfficeInterface.ObjectInfo.Description
MOIOfficeInterface.ObjectInfo.ExecuteAndDisplayDefaultInExcel
MOIOfficeInterface.ObjectInfo.ExecuteAndDisplayDefaultInPowerPoint
MOIOfficeInterface.ObjectInfo.ExecuteAndDisplayDefaultInWord
MOIOfficeInterface.ObjectInfo.ExecuteAndDisplayInExcel
MOIOfficeInterface.ObjectInfo.ExecuteAndDisplayInPowerPoint
MOIOfficeInterface.ObjectInfo.ExecuteAndDisplayInWord
MOIOfficeInterface.ObjectInfo.ExecuteDocument
MOIOfficeInterface.ObjectInfo.ExecuteReport
MOIOfficeInterface.ObjectInfo.ExecuteRSDocument
MOIOfficeInterface.ObjectInfo.FolderID
MOIOfficeInterface.ObjectInfo.IsHidden
MOIOfficeInterface.ObjectInfo.ModificationTime
MOIOfficeInterface.ObjectInfo.ObjectID
MOIOfficeInterface.ObjectInfo.ObjectName
MOIOfficeInterface.ObjectInfo.Owner
MOIOfficeInterface.ObjectInfo.Path
MOIOfficeInterface.ObjectInfo.SubType
MOIOfficeInterface.ObjectInfo.Type
MOIOfficeInterface.ObjectInfos.Count
MOIOfficeInterface.ObjectInfos.Item
MOIOfficeInterface.Project.Connect
MOIOfficeInterface.Project.Disconnect
MOIOfficeInterface.Project.DocumentObjectTemplatesFolderID
MOIOfficeInterface.Project.DocumentObjectTemplatesFolderPath
MOIOfficeInterface.Project.GetAllReports
MOIOfficeInterface.Project.GetAttributeElements
MOIOfficeInterface.Project.GetElements
MOIOfficeInterface.Project.GetFolderContents
MOIOfficeInterface.Project.GetHistoryList
MOIOfficeInterface.Project.MyReportsFolderID
MOIOfficeInterface.ProjectMyReportsFolderPath
MOIOfficeInterface.Project.ProjectID
MOIOfficeInterface.Project.ProjectName
MOIOfficeInterface.Project.ReportObjectTemplatesFolderID
MOIOfficeInterface.Project.ReportObjectTemplatesFolderPath
MOIOfficeInterface.Project.RootFolderID
MOIOfficeInterface.Project.SearchObjects
MOIOfficeInterface.Project.SharedReportsFolderID
MOIOfficeInterface.Project.SharedReportsFolderPath
MOIOfficeInterface.Projects.Count
MOIOfficeInterface.Projects.Item
MOIOfficeInterface.ProjectSource.AuthenticationMode
MOIOfficeInterface.ProjectSource.DisconnectProjects
MOIOfficeInterface.ProjectSource.GetProjects
MOIOfficeInterface.ProjectSource.GetProjectsWithAuthMode
MOIOfficeInterface.ProjectSource.ProjectSourceName
MOIOfficeInterface.ProjectSource.ServerName
MOIOfficeInterface.ProjectSources.Count
MOIOfficeInterface.ProjectSources.Item
MOIOfficeInterface.Prompt.GetAttrExpressionPrompt
MOIOfficeInterface.Prompt.GetHierarchyPrompt
MOIOfficeInterface.Prompt.GetLevelPrompt
MOIOfficeInterface.Prompt.GetMetricExpressionPrompt
MOIOfficeInterface.Prompt.GetObjectPrompt
MOIOfficeInterface.Prompt.GetPromptElements
MOIOfficeInterface.Prompt.GetSAPExpressionPrompt
MOIOfficeInterface.Prompt.GetSimplePrompt
MOIOfficeInterface.Prompt.HasDefaultAnswer
MOIOfficeInterface.Prompt.Maximum
MOIOfficeInterface.Prompt.Meaning
MOIOfficeInterface.Prompt.Minimum
MOIOfficeInterface.Prompt.Name
MOIOfficeInterface.Prompt.PromptID
MOIOfficeInterface.Prompt.PromptIndex
MOIOfficeInterface.Prompt.PromptType
MOIOfficeInterface.Prompt.Required
MOIOfficeInterface.Prompt.Title
MOIOfficeInterface.PromptAttrExpression.GetAttributeAnswer
MOIOfficeInterface.PromptAttrExpression.GetAvailableAttributes
MOIOfficeInterface.PromptAttrExpression.SetAttributeAnswer
MOIOfficeInterface.PromptAttributeAnswer.AnswerValue1
MOIOfficeInterface.PromptAttributeAnswer.AnswerValue2
MOIOfficeInterface.PromptAttributeAnswer.GetAttribute
MOIOfficeInterface.PromptAttributeAnswer.GetQualifier
MOIOfficeInterface.PromptAttributeAnswer.etSelectedElements
MOIOfficeInterface.PromptAttributeAnswer.Operator
MOIOfficeInterface.PromptAttributeAnswer.SetAttribute
MOIOfficeInterface.PromptAttributeAnswer.SetQualifier
MOIOfficeInterface.PromptAttributeAnswer.SetSelectedElements
MOIOfficeInterface.PromptAttributeAnswers.Add
MOIOfficeInterface.PromptAttributeAnswers.Count
MOIOfficeInterface.PromptAttributeAnswers.Item
MOIOfficeInterface.PromptAttributeAnswers.Remove
MOIOfficeInterface.PromptAttributeInfo.AttributeID
MOIOfficeInterface.PromptAttributeInfo.AttributeUID
MOIOfficeInterface.PromptAttributeInfo.DisplayName
MOIOfficeInterface.PromptAttributeInfo.GetAvailableQualifiers
MOIOfficeInterface.PromptAttributeInfo.Type
MOIOfficeInterface.PromptAttributeInfos.Count
MOIOfficeInterface.PromptAttributeInfos.Item
MOIOfficeInterface.PromptElementInfo.ElementID
MOIOfficeInterface.PromptElementInfo.ElementNameCount
MOIOfficeInterface.PromptElementInfo.GetAttribute
MOIOfficeInterface.PromptElementInfo.GetElementNames
MOIOfficeInterface.PromptElementInfos.Add
MOIOfficeInterface.PromptElementInfos.Count
MOIOfficeInterface.PromptElementInfos.Item
MOIOfficeInterface.PromptElementInfos.Remove
MOIOfficeInterface.PromptHierarchy.DrillElementToAttribute
MOIOfficeInterface.PromptHierarchy.GetAttributeChildren
MOIOfficeInterface.PromptHierarchy.GetAttributeElements
MOIOfficeInterface.PromptHierarchy.GetAttributeParents
MOIOfficeInterface.PromptHierarchy.GetAvailableAttributes
MOIOfficeInterface.PromptHierarchy.GetAvailableHierarchies
MOIOfficeInterface.PromptHierarchy.GetFolderContents
MOIOfficeInterface.PromptHierarchy.GetHierarchyAnswers
MOIOfficeInterface.PromptHierarchy.GetHierarchyAttributes
MOIOfficeInterface.PromptHierarchy.SetAnswer
MOIOfficeInterface.PromptHierarchyAnswer.AnswerOperator
MOIOfficeInterface.PromptHierarchyAnswer.GetAttributeAnswer
MOIOfficeInterface.PromptHierarchyAnswer.SetAttributeAnswer
MOIOfficeInterface.PromptHierarchyAnswers.Add
MOIOfficeInterface.PromptHierarchyAnswers.Count
MOIOfficeInterface.PromptHierarchyAnswers.Item
MOIOfficeInterface.PromptHierarchyAnswers.Remove
MOIOfficeInterface.PromptLevel.GetAvailableLevelObjects
MOIOfficeInterface.PromptLevel.GetLevelAnswer
MOIOfficeInterface.PromptLevel.SetLevelAnswer
MOIOfficeInterface.PromptLevelInfo.AggregationType
MOIOfficeInterface.PromptLevelInfo.FilteringType
MOIOfficeInterface.PromptLevelInfo.GetPromptObject
MOIOfficeInterface.PromptLevelInfo.SetPromptObject
MOIOfficeInterface.PromptLevelInfos.Add
MOIOfficeInterface.PromptLevelInfos.Count
MOIOfficeInterface.PromptLevelInfos.Item
MOIOfficeInterface.PromptLevelInfos.Remove
MOIOfficeInterface.PromptMetricAnswer.AnswerValue1
MOIOfficeInterface.PromptMetricAnswer.AnswerValue2
MOIOfficeInterface.PromptMetricAnswer.Function
MOIOfficeInterface.PromptMetricAnswer.Metric
MOIOfficeInterface.PromptMetricAnswer.Operator
MOIOfficeInterface.PromptMetricAnswer.SetMetric
MOIOfficeInterface.PromptMetricExpression.GetAvailableMetrics
MOIOfficeInterface.PromptMetricExpression.GetMetricAnswer

MOIOfficeInterface.PromptMetricExpression.SetMetricAnswer
MOIOfficeInterface.PromptObject.GetAnswerObjects
MOIOfficeInterface.PromptObject.GetAttribute
MOIOfficeInterface.PromptObject.GetAttributeElements
MOIOfficeInterface.PromptObject.GetAvailableObjects
MOIOfficeInterface.PromptObject.IsObjectPrompt
MOIOfficeInterface.PromptObject.IsPredefinedList
MOIOfficeInterface.PromptObject.Restrictions
MOIOfficeInterface.PromptObject.SetAnswer
MOIOfficeInterface.PromptObjectInfo.Abbreviation
MOIOfficeInterface.PromptObjectInfo.Description
MOIOfficeInterface.PromptObjectInfo.DisplayName
MOIOfficeInterface.PromptObjectInfo.ObjectID
MOIOfficeInterface.PromptObjectInfo.ObjectUID
MOIOfficeInterface.PromptObjectInfo.State
MOIOfficeInterface.PromptObjectInfo.Subtype
MOIOfficeInterface.PromptObjectInfo.Type
MOIOfficeInterface.PromptObjectInfos.Add
MOIOfficeInterface.PromptObjectInfos.Count
MOIOfficeInterface.PromptObjectInfos.Item
MOIOfficeInterface.PromptObjectInfos.Remove
MOIOfficeInterface.PromptQualifierInfo.AttributeFormName
MOIOfficeInterface.PromptQualifierInfo.DataType
MOIOfficeInterface.PromptQualifierInfo.DisplayName
MOIOfficeInterface.PromptQualifierInfo.Initialize
MOIOfficeInterface.PromptQualifierInfo.QualifierID
MOIOfficeInterface.PromptQualifierInfo.QualifierUID
MOIOfficeInterface.PromptQualifierInfos.Count
MOIOfficeInterface.PromptQualifierInfos.Item
MOIOfficeInterface.Prompts.Count
MOIOfficeInterface.Prompts.Item
MOIOfficeInterface.PromptSAPExpression.AllowedOperator
MOIOfficeInterface.PromptSAPExpression.GetAvailableAttributes
MOIOfficeInterface.PromptSAPExpression.GetExcludingAttributeAnswers
MOIOfficeInterface.PromptSAPExpression.GetIncludingAttributeAnswers
MOIOfficeInterface.PromptSAPExpression.SetExcludingAttributeAnswers
MOIOfficeInterface.PromptSAPExpression.SetIncludingAttributeAnswers
MOIOfficeInterface.PromptSimple.Answer
MOIOfficeInterface.Report.ExecuteAndDisplayDefaultInExcel
MOIOfficeInterface.Report.ExecuteAndDisplayDefaultInPowerPoint
MOIOfficeInterface.Report.ExecuteAndDisplayDefaultInWord
MOIOfficeInterface.Report.ExecuteAndDisplayInExcel
MOIOfficeInterface.Report.ExecuteAndDisplayInPowerPoint
MOIOfficeInterface.Report.ExecuteAndDisplayInWord
MOIOfficeInterface.Report.ExecuteReport
MOIOfficeInterface.Report.Initialize
MOIOfficeInterface.ReportLocation.AnswerPrompt
MOIOfficeInterface.ReportLocation.CancelReport
MOIOfficeInterface.ReportLocation.DeleteReport
MOIOfficeInterface.ReportLocation.DisplayDefaultInExcel
MOIOfficeInterface.ReportLocation.DisplayDefaultInPowerPoint
MOIOfficeInterface.ReportLocation.DisplayDefaultInWord
MOIOfficeInterface.ReportLocation.DisplayReportInExcel
MOIOfficeInterface.ReportLocation.DisplayReportInPowerPoint
MOIOfficeInterface.ReportLocation.DisplayReportInWord
MOIOfficeInterface.ReportLocation.DisplayReportXMLInExcel
MOIOfficeInterface.ReportLocation.DisplayReportXMLInPowerPoint
MOIOfficeInterface.ReportLocation.DisplayReportXMLInWord
MOIOfficeInterface.ReportLocation.GetPrompts
MOIOfficeInterface.ReportLocation.GetReportResults
MOIOfficeInterface.ReportLocation.PersistReport
MOIOfficeInterface.ReportLocation.RefreshReport
MOIOfficeInterface.ReportLocation.ReportLocationID
MOIOfficeInterface.ReportLocation.ResultHTML
MOIOfficeInterface.ReportLocation.ResultXml
MOIOfficeInterface.ReportLocations.Count
MOIOfficeInterface.ReportLocations.Item
MOIOfficeInterface.Reports.Count
MOIOfficeInterface.Reports.Item
MOIOfficeInterface.Restriction.Flag
MOIOfficeInterface.Restriction.Value
MOIOfficeInterface.Restrictions.Add
MOIOfficeInterface.Restrictions.Count
MOIOfficeInterface.Restrictions.Item
MOIOfficeInterface.Restrictions.Remove
MOIOfficeInterface.RSDocument.ExecuteAndDisplayDefaultInExcel
MOIOfficeInterface.RSDocument.ExecuteRSDocument
MOIOfficeInterface.RSDocument.Initialize
MOIOfficeInterface.RSDocuments.Count
MOIOfficeInterface.RSDocuments.Item
MOIOfficeInterface.RSLocation.AnswerPrompt
MOIOfficeInterface.RSLocation.CancelRSDocument
MOIOfficeInterface.RSLocation.DeleteRSDocument
MOIOfficeInterface.RSLocation.DisplayDefaultInExcel
MOIOfficeInterface.RSLocation.DisplayRSDocumentInExcel
MOIOfficeInterface.RSLocation.GetPrompts
MOIOfficeInterface.RSLocation.GetReportResults
MOIOfficeInterface.RSLocation.PersistRSDocument
MOIOfficeInterface.RSLocation.RefreshRSDocument
MOIOfficeInterface.RSLocation.RSLocationID

Intelligence Server API

IDSSAccessControlEntry
IDSSAccessControlEntry::DeniesAccess
IDSSAccessControlEntry::Inheritable
IDSSAccessControlEntry::Rights
IDSSAccessControlEntry::Trustee
IDSSAccessControlEntry::Type
IDSSAccessControlList
IDSSAccessControlList::_NewEnum
IDSSAccessControlList::Add
IDSSAccessControlList::Clear
IDSSAccessControlList::Compress
IDSSAccessControlList::ComputeUserRights
IDSSAccessControlList::Count
IDSSAccessControlList::Item
IDSSAccessControlList::PropagateACL
IDSSAccessControlList::Remove
IDSSAEInfo
IDSSAEInfo::get_AECalculationInfo
IDSSAggMetric
IDSSAggMetric::AllChildrenJoin
IDSSAggMetric::Column
IDSSAggMetric::Expression
IDSSAggMetric::Info
IDSSAggMetric::Populate
IDSSAggMetric::Root
IDSSApplicationServices
IDSSApplicationServices::FindDependentColumns
IDSSApplicationServices::PropertyHelper
IDSSAttribute
IDSSAttribute::Ancestors
IDSSAttribute::Arity
IDSSAttribute::AttributeType
IDSSAttribute::BaseAttribute
IDSSAttribute::BrowseForms
IDSSAttribute::BrowsePath
IDSSAttribute::Children
IDSSAttribute::DefaultChild
IDSSAttribute::DefaultParent
IDSSAttribute::Definition
IDSSAttribute::Descendents
IDSSAttribute::Dimension
IDSSAttribute::DrillMap
IDSSAttribute::ElementFilters
IDSSAttribute::Elements
IDSSAttribute::FetchBottomForms
IDSSAttribute::Forms
IDSSAttribute::GetFilteredElements
IDSSAttribute::Info
IDSSAttribute::Level
IDSSAttribute::LockLimit
IDSSAttribute::LockReason
IDSSAttribute::LockType
IDSSAttribute::Parents
IDSSAttribute::PartialType

IDSSAttribute::Populate
IDSSAttribute::Role
IDSSAttribute::Sorts
IDSSAttribute::TemplateForms
IDSSAttribute::Transformation
IDSSAttribute::Weight
IDSSAttribute2
IDSSAttribute2::AllForms
IDSSAttribute2::LocalBrowseForms
IDSSAttribute2::LocalTemplateForms
IDSSAttribute2::LookupForm
IDSSAttribute2::NewWrapper
IDSSAttribute2::UnusedBaseFormID
IDSSAttribute2::Wrapper
IDSSAttribute2::XDAForms
IDSSAttributeDefn
IDSSAttributeDefn::BaseForms
IDSSAttributeDefn::KeyColumnSize
IDSSAttributeDefn2
IDSSAttributeDefn2::AllBaseForms
IDSSAttributeDefn2::LookupBaseForm
IDSSAttributeDefn2::XDABaseForms
IDSSAttributeForm
IDSSAttributeForm::_NewEnum
IDSSAttributeForm::Add
IDSSAttributeForm::BaseAttributeForm
IDSSAttributeForm::Clear
IDSSAttributeForm::Count
IDSSAttributeForm::Info
IDSSAttributeForm::Item
IDSSAttributeForm::Populate
IDSSAttributeForm::Remove
IDSSAttributeForm::Type
IDSSAttributeForm::UseDescription
IDSSAttributeForm::UseName
IDSSAttributeForms
IDSSAttributeForms::_NewEnum
IDSSAttributeForms::Add
IDSSAttributeForms::BaseAttributeForms
IDSSAttributeForms::Clear
IDSSAttributeForms::Count
IDSSAttributeForms::Item
IDSSAttributeForms::Remove
IDSSAttributeInfo
IDSSAttributeInfo::_NewEnum
IDSSAttributeInfo::Add
IDSSAttributeInfo::Attribute
IDSSAttributeInfo::Clear
IDSSAttributeInfo::Count
IDSSAttributeInfo::Default
IDSSAttributeInfo::Item
IDSSAttributeInfo::LocalDefn
IDSSAttributeInfo::PartialAttributeID
IDSSAttributeInfo::PartialType
IDSSAttributeInfo::Remove
IDSSAttributeInfo2
IDSSAttributeInfo2::AttributeForms
IDSSAttributeInfo2::BrowseForms
IDSSAttributeInfo2::Column
IDSSAttributeInfo2::Populate
IDSSAttributeInfo2::Prune
IDSSAttributeInfo3
IDSSAttributeInfo3::GetAttributeFormsWith Restrictions
IDSSAttributeInfo3::PruneWithRestrictions
IDSSAttributeInfos
IDSSAttributeInfos::_NewEnum
IDSSAttributeInfos::Add
IDSSAttributeInfos::Clear
IDSSAttributeInfos::Count
IDSSAttributeInfos::Item
IDSSAttributeInfos::Remove
IDSSAttributeRelationship
IDSSAttributeRelationship::_NewEnum
IDSSAttributeRelationship::Attribute
IDSSAttributeRelationship::Count
IDSSAttributeRelationship::ElementsCollection
IDSSAttributeRelationship::HasRelationship
IDSSAttributeRelationship::Item
IDSSAttributeRelationship::Relationship
IDSSAttributeRelationships
IDSSAttributeRelationships::_NewEnum
IDSSAttributeRelationships::Add
IDSSAttributeRelationships::Clear
IDSSAttributeRelationships::Count
IDSSAttributeRelationships::Item
IDSSAttributeRelationships::Remove
IDSSAttributes
IDSSAttributes::_NewEnum
IDSSAttributes::Add
IDSSAttributes::Clear
IDSSAttributes::Count
IDSSAttributes::Item
IDSSAttributes::Remove
IDSSAttributeWrapper
IDSSAttributeWrapper::Attribute
IDSSAttributeWrapper::DefinitionTable
IDSSAttributeWrapper:: RelationshipsDimension
IDSSAttributeWrapper::WrapperAsAttribute
IDSSAxes
IDSSAxes::_NewEnum
IDSSAxes::Add
IDSSAxes::Count
IDSSAxes::Item
IDSSAxes::Remove
IDSSAxis
IDSSAxis::_NewEnum
IDSSAxis::Add
IDSSAxis::AddCopy
IDSSAxis::Axes
IDSSAxis::AxisElements
IDSSAxis::Clear
IDSSAxis::Count
IDSSAxis::Index
IDSSAxis::Item
IDSSAxis::Prompt
IDSSAxis::Remove
IDSSAxis::Sorts
IDSSAxis2
IDSSAxis2::GridFormat
IDSSAxis2::HeaderFormat
IDSSAxis2::SubtotalGridFormat
IDSSAxis2::SubtotalHeaderFormat
IDSSAxis3
IDSSAxis3::DisabledCount
IDSSAxis3::DisabledItem
IDSSBaseAttributeForm
IDSSBaseAttributeForm::BaseAttribute
IDSSBaseAttributeForm::BaseFormID
IDSSBaseAttributeForm::BaseFormType
IDSSBaseAttributeForm::Column
IDSSBaseAttributeForm::Default
IDSSBaseAttributeForm::Description
IDSSBaseAttributeForm::Expression
IDSSBaseAttributeForm::Name
IDSSBaseAttributeForm::Root
IDSSBaseAttributeForm::Table
IDSSBaseAttributeForms
IDSSBaseAttributeForms::_NewEnum
IDSSBaseAttributeForms::Add
IDSSBaseAttributeForms::Clear
IDSSBaseAttributeForms::Count
IDSSBaseAttributeForms::Item
IDSSBaseAttributeForms::Remove
IDSSBigDecimal
IDSSBigDecimal::AsDouble
IDSSBigDecimal::AsFormattedString
IDSSBigDecimal::AsFormattedStringFrom Handle
IDSSBigDecimal::AsLong
IDSSBigDecimal::AsRawString
IDSSBigDecimal::CompareTo
IDSSBigDecimal::Precision
IDSSBigDecimal::Scale
IDSSBigDecimal::Signum
IDSSBigDecimalFactory
IDSSBigDecimalFactory::CreateFromDouble
IDSSBigDecimalFactory::CreateFromInteger
IDSSBigDecimalFactory::CreateFromString
IDSSBigDecimalNode
IDSSBigDecimalNode::BigDecimal
IDSSBigDecimalNode::Node
IDSSBigDecimalNode::ParseString
IDSSBigDecimalNode::Prompt
IDSSBigDecimalNode::SetFromBigDecimal
IDSSBigDecimalNode::SetFromDouble
IDSSBigDecimalNode::SetFromLong
IDSSBigDecimalNode::SetFromRawString
IDSSBindingTable
IDSSBindingTable::_NewEnum
IDSSBindingTable::Add
IDSSBindingTable::AddDependents
IDSSBindingTable::Clear
IDSSBindingTable::Count
IDSSBindingTable::Find
IDSSBindingTable::Item
IDSSBindingTable::Position
IDSSBindingTable::Remove
IDSSBindingTable::Update
IDSSBindingTable2
IDSSBindingTable2::FindByID
IDSSCacheAdmin
IDSSCacheAdmin::CacheTableBucketCount
IDSSCacheAdmin::CacheTableBucketPerLock
IDSSCacheAdmin::FileCachePath
IDSSCacheAdmin::GetGeneralCacheInfo
IDSSCacheAdmin::MaxCacheCount
IDSSCacheAdmin::MaxMemoryConsumption
IDSSCacheAdmin::SwapPolicy
IDSSCacheFileObjectSource
IDSSCacheFileObjectSource::ContentSource
IDSSCatalog
IDSSCatalog::Columns
IDSSCatalog::Definition
IDSSCatalog::Info
IDSSCatalog::Tables
IDSSCatalog::TableSources
IDSSCatalog::Warehouse
IDSSCatalog::WHTables
IDSSCatalog2
IDSSCatalog2::XDADefinition
IDSSCatalog2::XDAType
IDSSCatalogDefn
IDSSCatalogDefn::CatalogSQL
IDSSCatalogDefn::Driver
IDSSCatalogDefn::Info
IDSSCatalogDefn::Restrict
IDSSCatalogDefn::TableSQL
IDSSCatalogDefn::Update
IDSSCatalogDefn::UpdateTables
IDSSCatalogDefn2
IDSSCatalogDefn2::ColumnSQL
IDSSCatalogDefn3
IDSSCatalogDefn3::DBLogin
IDSSChannel

IDSSChannel::MaxConnection
IDSSChannel::Name
IDSSChannel::port
IDSSChannel::Type
IDSSChannels
IDSSChannels::_NewEnum
IDSSChannels::Add
IDSSChannels::Clear
IDSSChannels::Count
IDSSChannels::Item
IDSSChannels::Remove
IDSSClientServices
IDSSClientServices::UserAcctSvcs
IDSSClientServices2
IDSSClientServices2::Auditor
IDSSClientServices3
IDSSClientServices3::ClientServicesInstance
IDSSClientServices3::ServerDefManipulator
IDSSClientServices3::UserFullPrivileges
IDSSClientServices4
IDSSClientServices4::ObjectManipulator
IDSSClientServices4::PromptEvaluator
IDSSClusterManager
IDSSClusterManager::GetClusterMembers
IDSSClusterManager::JoinCluster
IDSSClusterManager::LeaveCluster
IDSSClusterManager::Name
IDSSClusterManager::RefreshClusterMember
IDSSClusterManager::RemoveFromCluster
IDSSClusterManager2
IDSSClusterManager2::GetLoadedProjects OnServers
IDSSClusterManager2::GetProjectLevel ClusteringPolicyProjectMapping
IDSSClusterManager2::GetProjectLevel ClusteringPolicyProjectRedundency
IDSSClusterManager2::GetProjectStatus OnServers
IDSSClusterManager2::PerformProject LoadBalancing
IDSSClusterManager2::ProjectFailoverLatency
IDSSClusterManager2::ProjectLevelClustering
IDSSClusterManager2:: ProjectMigrationLatency
IDSSClusterManager2::SetProjectLevel ClusteringPolicyProjectMapping
IDSSClusterManager2::SetProjectLevel ClusteringPolicyProjectRedundency
IDSSClusterManager3
IDSSClusterManager3::GetClusteringStartup Membership
IDSSClusterManager3::SetClusteringStartup Membership
IDSSClusterMember
IDSSClusterMember::ContactPort
IDSSClusterMember::ID
IDSSClusterMember::Load
IDSSClusterMember::Protocol
IDSSClusterMember::Status
IDSSClusterMembers
IDSSClusterMembers::_NewEnum
IDSSClusterMembers::Count
IDSSClusterMembers::Item
IDSSClusterMembers::ItemById
IDSSCollection
IDSSCollection::_NewEnum
IDSSCollection::Add
IDSSCollection::Clear
IDSSCollection::Count
IDSSCollection::Item
IDSSCollection::Remove
IDSSColumn
IDSSColumn::ColumnName
IDSSColumn::ColumnType
IDSSColumn::DataType
IDSSColumn::Info
IDSSColumn::Missing
IDSSColumn::Populate
IDSSColumn2
IDSSColumn2::XDADefinition
IDSSColumn2::XDAType
IDSSColumns
IDSSColumns::_NewEnum
IDSSColumns::Add
IDSSColumns::Clear
IDSSColumns::Count
IDSSColumns::Item
IDSSColumns::Remove
IDSSCommand
IDSSCommand::CommandDescription
IDSSCommand::CommandID
IDSSCommand::ComponentProgID
IDSSCommand::Name
IDSSCommand::ParameterCount
IDSSCommand::ParameterNames
IDSSCommand::ParameterTypes
IDSSCommands
IDSSCommands::_NewEnum
IDSSCommands::Add
IDSSCommands::Clear
IDSSCommands::Count
IDSSCommands::Item
IDSSCommands::Remove
IDSSComments
IDSSComments::_NewEnum
IDSSComments::Add
IDSSComments::Count
IDSSComments::Item
IDSSComments::Remove
IDSSComponent
IDSSComponent::CleanUp
IDSSComponent::Init
IDSSComponent::Role
IDSSConditionality
IDSSConditionality::EmbedMethod
IDSSConditionality::Filter
IDSSConditionality::FilterExports
IDSSConditionality::FilterPrompt
IDSSConditionality::RemoveRelatedReport FilterElements
IDSSConfiguration
IDSSConfiguration::Info
IDSSConfiguration::Populate
IDSSConfiguration::Root
IDSSConfiguration::SystemPropertySet
IDSSConfiguration2
IDSSConfiguration2::ManagedObjects
IDSSConsolidation
IDSSConsolidation::DrillMap
IDSSConsolidation::Elements
IDSSConsolidation::ExtendedType
IDSSConsolidation::FindLeafElements
IDSSConsolidation::FindRelatedElements
IDSSConsolidation::Info
IDSSConsolidation::Populate
IDSSConsolidation::Prompt
IDSSConsolidation2
IDSSConsolidation2::Aggregation
IDSSConsolidationElement
IDSSConsolidationElement::Expression
IDSSConsolidationElement::ExtendedType
IDSSConsolidationElement::Info
IDSSConsolidationElement::Populate
IDSSConsolidationElement::Root
IDSSConsolidationElement2
IDSSConsolidationElement2::ChildGridFormat
IDSSConsolidationElement2:: ChildHeaderFormat
IDSSConsolidationElement2::GridFormat
IDSSConsolidationElement2::HeaderFormat
IDSSConstant
IDSSConstant::DataType
IDSSConstant::Node
IDSSConstant::Prompt
IDSSConstant::Text
IDSSConstant::Value
IDSSConstant2
IDSSConstant2::IsNull
IDSSContentServer
IDSSContentServer::ContentSource
IDSSContentSource
IDSSContentSource::Administrator
IDSSContentSource::CopyObject
IDSSContentSource::DeleteObject
IDSSContentSource::ExecuteSearch
IDSSContentSource::FindObject
IDSSContentSource::LoadSchema
IDSSContentSource::MoveObject
IDSSContentSource::NewObject
IDSSContentSource::RefreshObject
IDSSContentSource::SaveObject
IDSSContentSource::Separator
IDSSContractInfo
IDSSContractInfo::Context
IDSSContractInfo::Handle
IDSSContractInfo::Size
IDSSContractInfo::Source
IDSSContractInfo::TimeStamp
IDSSContractInfoCollection
IDSSContractInfoCollection::_NewEnum
IDSSContractInfoCollection::Count
IDSSContractInfoCollection::Item
IDSSContractManager
IDSSContractManager::CancelContract
IDSSContractManager::GetAvailable
IDSSContractManager::GetContractedTotal
IDSSContractManager::GetContractInfo
IDSSContractManager::GetContractList
IDSSContractManager::InitiateContract
IDSSContractManager::InitiateMinContract
IDSSContractManager::SetMaxContractPolicy
IDSSContractManager::SetMCMWatermark
IDSSContractManager:: SetMemoryRequestIdleTime
IDSSContractManager::SetMemoryReserve
IDSSContractManager:: SetMemoryReservePercent
IDSSContractManager::SetTotalLimitPolicy
IDSSContractManager::SetVirtualByteLimit Percent
IDSSContractManager::UpdateContract
IDSSCreate
IDSSCreate::Create
IDSSCustomGroup
IDSSCustomGroup::DrillMap
IDSSCustomGroup::Expression
IDSSCustomGroup::ExtendedType
IDSSCustomGroup::FilterType
IDSSCustomGroup::Info
IDSSCustomGroup::Populate
IDSSCustomGroup::Root
IDSSCustomGroup2
IDSSCustomGroup2::Aggregation
IDSSCustomGroup2::Flatten
IDSSCustomGroup3
IDSSCustomGroup3::IsParentFirst

IDSSCustomSQL
IDSSCustomSQL::Expression
IDSSCustomSQL::Text
IDSSCustomSQL::XDADBTable
IDSSCustomSQLColumn
IDSSCustomSQLColumn::SelectListIndex
IDSSCustomSQLColumn::XDAColumn
IDSSCustomSQLQueryBuilder
IDSSCustomSQLQueryBuilder::BuildJoinTree
IDSSCustomSQLQueryBuilder::Having
IDSSCustomSQLQueryBuilder::JoinList
IDSSCustomSQLQueryBuilder::
JoinOrderIgnored
IDSSCustomSQLQueryBuilder::JoinTree
IDSSCustomSQLQueryBuilder::QBColumns
IDSSCustomSQLQueryBuilder::
QBQueryColumns
IDSSCustomSQLQueryBuilder::QBTables
IDSSCustomSQLQueryBuilder::Where
IDSSCustomSubtotal
IDSSCustomSubtotal::_NewEnum
IDSSCustomSubtotal::Add
IDSSCustomSubtotal::Clear
IDSSCustomSubtotal::Count
IDSSCustomSubtotal::Item
IDSSCustomSubtotal::Remove
IDSSCustomSubtotal::TemplateSubtotal
IDSSCustomSubtotals
IDSSCustomSubtotals::_NewEnum
IDSSCustomSubtotals::Add
IDSSCustomSubtotals::Clear
IDSSCustomSubtotals::Count
IDSSCustomSubtotals::Item
IDSSCustomSubtotals::Remove
IDSSDatabaseAdministrator
IDSSDatabaseAdministrator::
DatabaseMonitors
IDSSDatabaseAdministrator::DBRoleMonitors
IDSSDatabaseMonitor
IDSSDatabaseMonitor::ConnectionCount
IDSSDatabaseMonitor::CurrentLoad
IDSSDatabaseMonitor::DBConnection
IDSSDatabaseMonitor::DBConnectionID
IDSSDatabaseMonitor::DBConnectionName
IDSSDatabaseMonitor::MaxConnectionCount
IDSSDatabaseMonitor::MaximumLoad
IDSSDatabaseMonitor::Status
IDSSDatabaseMonitors
IDSSDatabaseMonitors::_NewEnum
IDSSDatabaseMonitors::Count
IDSSDatabaseMonitors::ItemByIndex
IDSSDatabaseMonitors::ItemByName
IDSSDataElement
IDSSDataElement::DerivedMetrics
IDSSDataElement::IsGroupByElementSource
IDSSDataElement::Object
IDSSDataElements
IDSSDataElements::_NewEnum
IDSSDataElements::Add
IDSSDataElements::Clear
IDSSDataElements::Count
IDSSDataElements::Default
IDSSDataElements::Item
IDSSDataElements::Remove
IDSSDatamart
IDSSDatamart::_NewEnum
IDSSDatamart::Add
IDSSDatamart::Clear
IDSSDatamart::Count
IDSSDatamart::DBRole
IDSSDatamart::Info
IDSSDatamart::IsPattern
IDSSDatamart::Item
IDSSDatamart::Populate
IDSSDatamart::Remove
IDSSDatamart::TableName
IDSSDatamartColumn
IDSSDatamartColumn::Column
IDSSDatamartColumn::Index
IDSSDatamartColumn::IsPattern
IDSSDatamartColumn::Name
IDSSDatamartColumn::Type
IDSSDatamartReport
IDSSDatamartReport::_NewEnum
IDSSDatamartReport::Add
IDSSDatamartReport::Clear
IDSSDatamartReport::Count
IDSSDatamartReport::Execute
IDSSDatamartReport::Info
IDSSDatamartReport::Item
IDSSDatamartReport::Populate
IDSSDatamartReport::Remove
IDSSDatamartServer
IDSSDatamartServer::CancelDatamartRequest
IDSSDatamartServer::Execute
IDSSDatamartServer::GetExecutionStatus
IDSSDatamartServer::GetInBoxMsg
IDSSDatamartUnit
IDSSDatamartUnit::Datamart
IDSSDatamartUnit::MakeTable
IDSSDatamartUnit::MapToColumn
IDSSDatamartUnit::MapToIndex
IDSSDatamartUnit::MapToTarget
IDSSDatamartUnit::Table
IDSSDatamartUnit::UseAppend
IDSSDatamartUnit::Validate
IDSSDataSource
IDSSDataSource::Administrator
IDSSDataSource::ApplicationType
IDSSDataSource::AuthMode
IDSSDataSource::AuthModesAvailable
IDSSDataSource::CacheSetting
IDSSDataSource::ConfigurationID
IDSSDataSource::ConfigurationName
IDSSDataSource::Create
IDSSDataSource::CreateSession
IDSSDataSource::DBType
IDSSDataSource::Delete
IDSSDataSource::DeleteFromRegistry
IDSSDataSource::Duplicate
IDSSDataSource::Enumerator
IDSSDataSource::Flags
IDSSDataSource::Hidden
IDSSDataSource::Init
IDSSDataSource::InternalProjectID
IDSSDataSource::IsInitialized
IDSSDataSource::Locale
IDSSDataSource::Location
IDSSDataSource::login
IDSSDataSource::MDLogin
IDSSDataSource::MDPasswd
IDSSDataSource::MDVersion
IDSSDataSource::Mode
IDSSDataSource::Name
IDSSDataSource::NewPasswd
IDSSDataSource::passwd
IDSSDataSource::port
IDSSDataSource::ProjectDescription
IDSSDataSource::ProjectID
IDSSDataSource::ProjectName
IDSSDataSource::Reset
IDSSDataSource::SaveToRegistry
IDSSDataSource::ServerAdmin
IDSSDataSource::ServerInstanceID
IDSSDataSource::ServerInstanceName
IDSSDataSource::SessionFlags
IDSSDataSource::Timeout
IDSSDataSource::Type
IDSSDataSource::WebServerAdmin
IDSSDataSource::WHLogin
IDSSDataSource::WHPasswd
IDSSDataSource2
IDSSDataSource2::Populate
IDSSDataSource2::ServerVersion
IDSSDataSource3
IDSSDataSource3::ClearKeyValues
IDSSDataSource3::KeyValue
IDSSDataSource3::RemoveKeyValue
IDSSDataSource4
IDSSDataSource4::IsNoMDPassword
IDSSDataSourceEnumerator
IDSSDataSourceEnumerator::_NewEnum
IDSSDataSourceEnumerator::Add
IDSSDataSourceEnumerator::AddDataSource
IDSSDataSourceEnumerator::Count
IDSSDataSourceEnumerator::Item
IDSSDataSourceEnumerator::
ItemByProjectName
IDSSDataSourceEnumerator::LoadFromRegistry
IDSSDataSourceEnumerator::
RemoveDataSource
IDSSDataSourceEnumerator::Reset
IDSSDataSourceEnumerator::SaveToRegistry
IDSSDataType
IDSSDataType::BitLength
IDSSDataType::ByteLength
IDSSDataType::DecimalScale
IDSSDataType::get_DataType
IDSSDataType::Precision
IDSSDataType::put_DataType
IDSSDataType::TimeScale
IDSSDataType::Type
IDSSDBConnection
IDSSDBConnection::CacheOption
IDSSDBConnection::CacheSize
IDSSDBConnection::CacheTimeOut
IDSSDBConnection::ConnectionString
IDSSDBConnection::ConnectionTimeOut
IDSSDBConnection::DatabaseName
IDSSDBConnection::DatabaseType
IDSSDBConnection::DatabaseVersion
IDSSDBConnection::DBCPrefixString
IDSSDBConnection::DefaultDBLogin
IDSSDBConnection::
DefaultMaxDBConnection
IDSSDBConnection::DefaultMaxDBCost
IDSSDBConnection::DriverType
IDSSDBConnection::ExecutionMode
IDSSDBConnection::Info
IDSSDBConnection::LoadDSNDefinition
IDSSDBConnection::MultiProcessOption
IDSSDBConnection::Populate
IDSSDBConnection::TablePrefixString
IDSSDBConnection::TableSpaceName
IDSSDBConnection::TraceFlag
IDSSDBConnection::UseExtendedFetch
IDSSDBConnectionMap
IDSSDBConnectionMap::DBConnection
IDSSDBConnectionMap::DBLogin
IDSSDBConnectionMap::DBRole
IDSSDBConnectionMap::User
IDSSDBConnectionMaps
IDSSDBConnectionMaps::_NewEnum
IDSSDBConnectionMaps::Add
IDSSDBConnectionMaps::Clear
IDSSDBConnectionMaps::Count

IDSSDBConnectionMaps::IsModified
IDSSDBConnectionMaps::Item
IDSSDBConnectionMaps::Items
IDSSDBConnectionMaps::Refresh
IDSSDBConnectionMaps::Remove
IDSSDBConnectionMaps::SaveChanges
IDSSDBConnections
IDSSDBConnections::_NewEnum
IDSSDBConnections::Add
IDSSDBConnections::Clear
IDSSDBConnections::Count
IDSSDBConnections::Item
IDSSDBConnections::Remove
IDSSDBElementServer
IDSSDBElementServer::Init
IDSSDBElementServer::Source
IDSSDBLogin
IDSSDBLogin::Info
IDSSDBLogin::LoginName
IDSSDBLogin::Password
IDSSDBLogin::Populate
IDSSDBMonitor
IDSSDBMonitor::Close
IDSSDBMonitor::ConnectionId
IDSSDBMonitor::CurrentCumulateCost
IDSSDBMonitor::DBConnection
IDSSDBMonitor::DBConnectionID
IDSSDBMonitor::DBConnectionName
IDSSDBMonitor::DBLogin
IDSSDBMonitor::DBLoginID
IDSSDBMonitor::DBLoginName
IDSSDBMonitor::DBRole
IDSSDBMonitor::DBRoleID
IDSSDBMonitor::DBRoleName
IDSSDBMonitor::EstimateTotalCost
IDSSDBMonitor::JobId
IDSSDBMonitor::JobStartTime
IDSSDBMonitor::PhyConnectionId
IDSSDBMonitor::SQL
IDSSDBMonitor::SQLStartTime
IDSSDBMonitor::Status
IDSSDBMonitor::User
IDSSDBMonitor::UserID
IDSSDBMonitor::UserName
IDSSDBMonitors
IDSSDBMonitors::_NewEnum
IDSSDBMonitors::Count
IDSSDBMonitors::Item
IDSSDBMonitors::ItemByIndex
IDSSDBMonitors2
IDSSDBMonitors2::
RemoveCachedDBConnection
IDSSDBMS
IDSSDBMS::DBMSType
IDSSDBMS::DBMSVersion
IDSSDBMS::Info
IDSSDBMS::MSIDescription
IDSSDBMS::MSIVersion
IDSSDBRole
IDSSDBRole::DatabaseName
IDSSDBRole::DatabaseType
IDSSDBRole::DatabaseVersion
IDSSDBRole::DBMS
IDSSDBRole::DefaultDBConnection
IDSSDBRole::DefaultDBConnections
IDSSDBRole::DefaultMaxDBConnection
IDSSDBRole::DefaultMaxDBCost
IDSSDBRole::Info
IDSSDBRole::Populate
IDSSDBRole::PrimaryDBRole
IDSSDBRole::Queue
IDSSDBRole::Tables
IDSSDBRole::TableSource
IDSSDBRole::TableSpaceName
IDSSDBRole2
IDSSDBRole2::SamePhysicalDatabase
IDSSDBRoleMonitor
IDSSDBRoleMonitor::ConnectionCount
IDSSDBRoleMonitor::CurrentLoad
IDSSDBRoleMonitor::DBRole
IDSSDBRoleMonitor::DBRoleID
IDSSDBRoleMonitor::DBRoleName
IDSSDBRoleMonitor::MaxConnectionCount
IDSSDBRoleMonitor::MaximumLoad
IDSSDBRoleMonitor::Status
IDSSDBRoleMonitors
IDSSDBRoleMonitors::_NewEnum
IDSSDBRoleMonitors::Count
IDSSDBRoleMonitors::ItemByIndex
IDSSDBRoleMonitors::ItemByName
IDSSDBRoles
IDSSDBRoles::_NewEnum
IDSSDBRoles::Add
IDSSDBRoles::Clear
IDSSDBRoles::Count
IDSSDBRoles::Item
IDSSDBRoles::Remove
IDSSDBTable
IDSSDBTable::Columns
IDSSDBTable::DBRole
IDSSDBTable::DbTableType
IDSSDBTable::Info
IDSSDBTable::Missing
IDSSDBTable::NameSpace
IDSSDBTable::Populate
IDSSDBTable::TableContent
IDSSDBTable::TableName
IDSSDBTable::TableSource
IDSSDBTable2
IDSSDBTable2::Keys
IDSSDBTable3
IDSSDBTable3::XDADefinition
IDSSDBTable3::XDAType
IDSSDBTableKey
IDSSDBTableKey::Columns
IDSSDBTableKey::Name
IDSSDBTableKey::References
IDSSDBTableKey::Type
IDSSDBTableKeys
IDSSDBTableKeys::_NewEnum
IDSSDBTableKeys::Add
IDSSDBTableKeys::Clear
IDSSDBTableKeys::Count
IDSSDBTableKeys::Item
IDSSDBTableKeys::Remove
IDSSDBTables
IDSSDBTables::_NewEnum
IDSSDBTables::Add
IDSSDBTables::Clear
IDSSDBTables::Count
IDSSDBTables::Item
IDSSDBTables::Remove
IDSSDelete
IDSSDelete::Delete
IDSSDependents
IDSSDependents::_NewEnum
IDSSDependents::Count
IDSSDependents::Item
IDSSDimension
IDSSDimension::_NewEnum
IDSSDimension::Add
IDSSDimension::Bottom
IDSSDimension::BrowseForms
IDSSDimension::Clear
IDSSDimension::Count
IDSSDimension::DefaultBottom
IDSSDimension::DefaultTop
IDSSDimension::DimensionType
IDSSDimension::DrillMap
IDSSDimension::Info
IDSSDimension::Item
IDSSDimension::Populate
IDSSDimension::Remove
IDSSDimension::TemplateForms
IDSSDimension::Top
IDSSDimension2
IDSSDimension2::NewWrapper
IDSSDimension2::Wrapper
IDSSDimensionInfo
IDSSDimensionInfo::Actual
IDSSDimensionInfo::Dimension
IDSSDimensionInfo::IsShared
IDSSDimensionInfo::PhysicalObjectIdentifier
IDSSDimensionInfo::RefreshStatus
IDSSDimensionInfo2
IDSSDimensionInfo2::Restrictions
IDSSDimensionInfos
IDSSDimensionInfos::_NewEnum
IDSSDimensionInfos::Add
IDSSDimensionInfos::Clear
IDSSDimensionInfos::Count
IDSSDimensionInfos::Item
IDSSDimensionInfos::Remove
IDSSDimensionMap
IDSSDimensionMap::DimCount
IDSSDimensions
IDSSDimensions::_NewEnum
IDSSDimensions::Add
IDSSDimensions::Clear
IDSSDimensions::Count
IDSSDimensions::Item
IDSSDimensions::Remove
IDSSDimensionWrapper
IDSSDimensionWrapper::DefinitionTable
IDSSDimensionWrapper::Dimension
IDSSDimensionWrapper::WrapperAsDimension
IDSSDimty
IDSSDimty::_NewEnum
IDSSDimty::Add
IDSSDimty::CanContinue
IDSSDimty::Clear
IDSSDimty::Count
IDSSDimty::FilterRest
IDSSDimty::Info
IDSSDimty::Item
IDSSDimty::Remove
IDSSDimtyUnit
IDSSDimtyUnit::_NewEnum
IDSSDimtyUnit::Add
IDSSDimtyUnit::Aggregation
IDSSDimtyUnit::Clear
IDSSDimtyUnit::Count
IDSSDimtyUnit::Filtering
IDSSDimtyUnit::GroupBy
IDSSDimtyUnit::Item
IDSSDimtyUnit::Key
IDSSDimtyUnit::RelativePosition
IDSSDimtyUnit::Remove
IDSSDimtyUnit::Role
IDSSDimtyUnit::Tag
IDSSDimtyUnit::Target
IDSSDimtyUnit::TargetPrompt
IDSSDimtyUnit::UnitType
IDSSDirectedAttribute
IDSSDirectedAttribute::Attribute
IDSSDirectedAttribute::Direction
IDSSDirectedAttributes
IDSSDirectedAttributes::_NewEnum
IDSSDirectedAttributes::Add

IDSSDirectedAttributes::Clear
IDSSDirectedAttributes::Count
IDSSDirectedAttributes::Item
IDSSDirectedAttributes::Remove
IDSSDocumentDefinition
IDSSDocumentDefinition::BaseDocument
IDSSDocumentDefinition::BaseDocumentID
IDSSDocumentDefinition::DocumentElements
IDSSDocumentDefinition::
DocumentElementsString
IDSSDocumentDefinition::DocumentType
IDSSDocumentDefinition::Info
IDSSDocumentDefinition::Layout
IDSSDocumentDefinition::LayoutString
IDSSDocumentDefinition::Populate
IDSSDocumentDefinition::PopulateFromXML
IDSSDocumentDefinition::
PopulateFromXMLString
IDSSDocumentDefinition::SaveToXML
IDSSDocumentDefinition::SaveToXMLString
IDSSDocumentDefinition2
IDSSDocumentDefinition2::AddRoot
IDSSDocumentDefinition2::ClearTree
IDSSDocumentDefinition2::CountNodes
IDSSDocumentDefinition2::DataElements
IDSSDocumentDefinition2::DefaultFormat
IDSSDocumentDefinition2::Node
IDSSDocumentDefinition2::PopulateTree
IDSSDocumentDefinition2::Resolution
IDSSDocumentDefinition2::Root
IDSSDocumentDefinition3
IDSSDocumentDefinition3::TotalSectionLimit
IDSSDocumentInstance
IDSSDocumentInstance::CancelExecution
IDSSDocumentInstance::Cookie
IDSSDocumentInstance::Definition
IDSSDocumentInstance::DocumentResolution
IDSSDocumentInstance::DocumentSource
IDSSDocumentInstance::Execute
IDSSDocumentInstance::ExecuteActions
IDSSDocumentInstance::ExecuteFlags
IDSSDocumentInstance::ExecutionStatus
IDSSDocumentInstance::HTMLInvalidate
IDSSDocumentInstance::HTMLOutput
IDSSDocumentInstance::ImagePath
IDSSDocumentInstance::JobID
IDSSDocumentInstance::LoadFromXMLDOM
IDSSDocumentInstance::Project
IDSSDocumentInstance::SaveToXMLDOM
IDSSDocumentInstance::State
IDSSDocumentInstance::UserData
IDSSDocumentInstance::UserRuntime
IDSSDocumentInstance::WriteGraphImages
IDSSDocumentInstance2
IDSSDocumentInstance2::ElementInBoxID
IDSSDocumentInstance2::GraphCount
IDSSDocumentInstance2::GraphImageData
IDSSDocumentInstance2::GridCount
IDSSDocumentInstance2::ImagePath2
IDSSDocumentInstance2::State2
IDSSDocumentInstance3
IDSSDocumentInstance3::CreateIterator
IDSSDocumentInstance3::ExportDocument
IDSSDocumentInstance3::ReportInstanceByID
IDSSDocumentInstance3::SaveAs
IDSSDocumentInstance3::UpdateState
IDSSDocumentInstance3::ValidateAndFix
IDSSDocumentInstance4
IDSSDocumentInstance4::ExportDocument2
IDSSDocumentInstance4:GroupByElementsTree
IDSSDocumentManipulator
IDSSDocumentManipulator::Accept
IDSSDocumentManipulator::
AddAttributeFormToField
IDSSDocumentManipulator::AddAttribute
FormToGroupByUnit
IDSSDocumentManipulator::AddDataElement
IDSSDocumentManipulator::AddDerivedMetric
IDSSDocumentManipulator::AddField
IDSSDocumentManipulator::AddGridForm
IDSSDocumentManipulator::
AddGroupBySection
IDSSDocumentManipulator::AddGroupByUnit
IDSSDocumentManipulator::AddMetric
IDSSDocumentManipulator::AddNode
IDSSDocumentManipulator::
AddPercentToTotalMetric
IDSSDocumentManipulator::AddRankMetric
IDSSDocumentManipulator::AddUnit
IDSSDocumentManipulator::ApplyDeltaXML
IDSSDocumentManipulator::ApplyObjectLock
IDSSDocumentManipulator::ApplyStyle
IDSSDocumentManipulator::AxisSubtotals
IDSSDocumentManipulator::Cancel
IDSSDocumentManipulator::
ChangeDisplayName
IDSSDocumentManipulator::
ClearDataElements
IDSSDocumentManipulator::ClearFields
IDSSDocumentManipulator::ClearNodes
IDSSDocumentManipulator::ClearSorts
IDSSDocumentManipulator::ClearTemplate
IDSSDocumentManipulator::CopyField
IDSSDocumentManipulator::
CopyTemplateNode
IDSSDocumentManipulator::EditField
IDSSDocumentManipulator::EditNode
IDSSDocumentManipulator::EditSectionNode
IDSSDocumentManipulator::FormatAxis
IDSSDocumentManipulator::FormatField
IDSSDocumentManipulator::FormatFieldXML
IDSSDocumentManipulator::
FormatGroupByUnit
IDSSDocumentManipulator::
FormatGroupByUnitXML
IDSSDocumentManipulator::FormatNode
IDSSDocumentManipulator::FormatNodeXML
IDSSDocumentManipulator::FormatUnit
IDSSDocumentManipulator::HideBanding
IDSSDocumentManipulator::HideSubtotals
IDSSDocumentManipulator::HideThresholds
IDSSDocumentManipulator::Locale
IDSSDocumentManipulator::
ManipulationType
IDSSDocumentManipulator::MoveField
IDSSDocumentManipulator::
MoveGroupBySection
IDSSDocumentManipulator::MoveNode
IDSSDocumentManipulator::
MoveTemplateNode
IDSSDocumentManipulator::PivotUnit
IDSSDocumentManipulator::
RemoveDataElement
IDSSDocumentManipulator::RemoveField
IDSSDocumentManipulator::RemoveGridForm
IDSSDocumentManipulator::
RemoveGroupBySection
IDSSDocumentManipulator::RemoveNode
IDSSDocumentManipulator::RemoveSubtotals
IDSSDocumentManipulator::RemoveUnit
IDSSDocumentManipulator::ResetGridForms
IDSSDocumentManipulator::
SetDefaultDataElement
IDSSDocumentManipulator::
SetDocumentPropertiesXML
IDSSDocumentManipulator::
SetGridDisplayMode
IDSSDocumentManipulator::SetNodeOption
IDSSDocumentManipulator::
SetPageThroughElement
IDSSDocumentManipulator::
SetTemplatePropertiesXML
IDSSDocumentManipulator::SetWidthClear
IDSSDocumentManipulator::SetWidthHeader
IDSSDocumentManipulator::SetWidthMetric
IDSSDocumentManipulator::SetWidthScenario
IDSSDocumentManipulator::Sort
IDSSDocumentManipulator::UnitSubtotals
IDSSDocumentManipulator::
UpdateDerivedMetric
IDSSDocumentManipulator2
IDSSDocumentManipulator2::AddCopy
TemplateNodeFromDataset
IDSSDocumentManipulator2::AddDerived
MetricToDataset
IDSSDocumentManipulator2::AddDerived
MetricToDataset2
IDSSDocumentManipulator2::AddField2
IDSSDocumentManipulator2::CutNodeTree
IDSSDocumentManipulator2::
FormatDerivedMetric
IDSSDocumentManipulator2::
FormatDerivedMetricXML
IDSSDocumentManipulator2::FormatTemplate
IDSSDocumentManipulator2::
HideDocumentThresholds
IDSSDocumentManipulator2::PasteNodeTree
IDSSDocumentManipulator2::RemoveDerived
MetricFromDataset
IDSSDocumentManipulator2::
SetControlDefaultFormat
IDSSDocumentManipulator2::
SetCurrentGroupByElement
IDSSDocumentManipulator2::
SetGraphPropertiesXML
IDSSDocumentManipulator2::SetNextCurrent
GroupByElement
IDSSDocumentManipulator2::SetPageByStyle
IDSSDocumentManipulator2::UpdateDerived
MetricOnDataset
IDSSDocumentManipulator3
IDSSDocumentManipulator3::SetViewFilter
IDSSDocumentNetClient
IDSSDocumentNetClient::CleanUp
IDSSDocumentNetClient::Execute
IDSSDocumentNetClient::ExecuteDefinition
IDSSDocumentNetClient::MsgCB
IDSSDocumentNetClient::NewInstance
IDSSDocumentNetClient::ProjectID
IDSSDocumentNetClient::PromptResume
IDSSDocumentNetClient::RemoteTimeOut
IDSSDocumentNetClient::Resolve
IDSSDocumentNetClient::Role
IDSSDocumentNetClient::ServerNet
IDSSDocumentServices
IDSSDocumentServices::Document
ManipulationService
IDSSDocumentSource
IDSSDocumentSource::Execute
IDSSDocumentSource::ExecuteDefinition
IDSSDocumentSource::NewInstance
IDSSDocumentSource::PromptResume
IDSSDocumentSource::Session
IDSSDocumentSourceEvents
IDSSDocumentSourceEvents::
ExecutionCanceled

IDSSDocumentSourceEvents::
ExecutionComplete
IDSSDocumentSourceEvents::ExecutionFailed
IDSSDocumentSourceEvents::
ProgressNotification
IDSSDocumentSourceEvents::ResolvePrompts
IDSSDocumentSourceServer
IDSSDocumentSourceServer::DocumentSource
IDSSDocumentSourceServer::PackageInstance
IDSSDocumentSourceServer::Process
IDSSDocumentSourceServer::Role
IDSSDocumentSourceServer::Source
IDSSDrillAction
IDSSDrillAction::Clear
IDSSDrillAction::Depth
IDSSDrillAction::DrillType
IDSSDrillAction::Filter
IDSSDrillAction::FilterMetrics
IDSSDrillAction::Forms
IDSSDrillAction::Inline
IDSSDrillAction::KeepOriginal
IDSSDrillAction::Name
IDSSDrillAction::NewObject
IDSSDrillAction::QualChange
IDSSDrillAction::Relationship
IDSSDrillAction::ReportFilter
IDSSDrillAction::SortChange
IDSSDrillAction::UserFilter
IDSSDrillAction::UseSchema
IDSSDrillAction2
IDSSDrillAction2::DrillAction
IDSSDrillAction2::DrillMetric
IDSSDrillAction2::DrillTemplateUnit
IDSSDrillAction2::DrillUnitSource
IDSSDrillAction2::HasDrillAction
IDSSDrillAction2::Index
IDSSDrillAction3
IDSSDrillAction3::KeepThresholds
IDSSDrillAction4
IDSSDrillAction4::KeepIndependent
IDSSDrillAction5
IDSSDrillAction5::DisplayMode
IDSSDrillMap
IDSSDrillMap::_NewEnum
IDSSDrillMap::Add
IDSSDrillMap::Clear
IDSSDrillMap::Count
IDSSDrillMap::Info
IDSSDrillMap::Item
IDSSDrillMap::Populate
IDSSDrillMap::Remove
IDSSDrillPath
IDSSDrillPath::Action
IDSSDrillPath::Description
IDSSDrillPath::Importance
IDSSDrillPath::Name
IDSSDrillPath::PathObject
IDSSDrillPath::PathType
IDSSDrillPath::SetName
IDSSDrillPath2
IDSSDrillPath2::IsGenerated
IDSSElement
IDSSElement::Ancestors
IDSSElement::Attribute
IDSSElement::Children
IDSSElement::ConsolidationElement
IDSSElement::Count
IDSSElement::DefaultChild
IDSSElement::DefaultChildElements
IDSSElement::DefaultParent
IDSSElement::Descendents
IDSSElement::Dimension
IDSSElement::ElementID
IDSSElement::ElementType
IDSSElement::GetFilteredElements
IDSSElement::GroupObject
IDSSElement::Item
IDSSElement::Metric
IDSSElement::Ordinal
IDSSElement::Parents
IDSSElement::SubtotalElements
IDSSElement::TemplateSubtotal
IDSSElement::TemplateUnit
IDSSElement::Text
IDSSElement::Value
IDSSElement::Width
IDSSElement2
IDSSElement2::ParentElement
IDSSElement2::ParentElementID
IDSSElementFilters
IDSSElementFilters::_NewEnum
IDSSElementFilters::Add
IDSSElementFilters::Clear
IDSSElementFilters::Count
IDSSElementFilters::Item
IDSSElementFilters::Remove
IDSSElementNetClient
IDSSElementNetClient::ElementSource
IDSSElementNetClient::Init
IDSSElementNetClient::RemoteTimeOut
IDSSElementNetClient::ServerNet
IDSSElementNetServer
IDSSElementNetServer::Process
IDSSElementReport
IDSSElementReport::ElementLimit
IDSSElementReport::ElementOffset
IDSSElementReport::ElementTotal
IDSSElements
IDSSElements::_NewEnum
IDSSElements::Add
IDSSElements::AddForms
IDSSElements::Clear
IDSSElements::Complete
IDSSElements::Count
IDSSElements::Filter
IDSSElements::Item
IDSSElements::Limit
IDSSElements::NewCollection
IDSSElements::Offset
IDSSElements::Refresh
IDSSElements::Remove
IDSSElements::Resolution
IDSSElementsCollection
IDSSElementsCollection::_NewEnum
IDSSElementsCollection::Complete
IDSSElementsCollection::Count
IDSSElementsCollection::Filter
IDSSElementsCollection::Item
IDSSElementsCollection::Limit
IDSSElementsCollection::NewCollection
IDSSElementsCollection::Offset
IDSSElementsCollection::Refresh
IDSSElementsCollection::Resolution
IDSSElementServer
IDSSElementServer::Init
IDSSElementServer::Source
IDSSElementsObject
IDSSElementsObject::Elements
IDSSElementsObject::Node
IDSSElementsObject::Populate
IDSSElementsObject::Prompt
IDSSElementSource
IDSSElementSource::GetElementFromID
IDSSElementSource::GetElements
IDSSElementSource::GetElementsXML
IDSSElementSource::GetHierarchyXML
IDSSElementSource::Purge
IDSSElementSource::RemoveAttribute
IDSSElementSource::RemoveElements
IDSSElementSource::RemoveForms
IDSSElementSource2
IDSSElementSource2::GetElementsEx
IDSSEmbeddedFolder
IDSSEmbeddedFolder::InUse
IDSSEmbeddedFolder::RemoveUnused
IDSSEmbeddedFolder::UsedBy
IDSSEmbeddedFolder::Uses
IDSSEmbeddedFolder2
IDSSEmbeddedFolder2::UsesExternal
IDSSEngineLite
IDSSEngineLite::Reduce
IDSSEngineLite::ReduceToHigher
IDSSEngineLite::ResetSchema
IDSSErrorInfo
IDSSErrorInfo::Description
IDSSErrorInfo::ErrorCode
IDSSErrors
IDSSErrors::_NewEnum
IDSSErrors::Add
IDSSErrors::Clear
IDSSErrors::Count
IDSSErrors::Item
IDSSErrors::Remove
IDSSExport
IDSSExport::ExportAs
IDSSExport::ExportIndex
IDSSExport::Prompt
IDSSExportContext
IDSSExportContext::Setting
IDSSExportEngineServer
IDSSExportEngineServer::ExportDocument
IDSSExportEngineServer::ExportReport
IDSSExportEngineServer::Initialize
IDSSExports
IDSSExports::_NewEnum
IDSSExports::Add
IDSSExports::Clear
IDSSExports::Count
IDSSExports::Item
IDSSExports::Remove
IDSSExpression
IDSSExpression::AddRoot
IDSSExpression::ClearExpression
IDSSExpression::Count
IDSSExpression::IsEquivalent
IDSSExpression::Item
IDSSExpression::Populate
IDSSExpression::Root
IDSSExpression::Text
IDSSExpression::TextEx
IDSSExpression2
IDSSExpression2::Populate2
IDSSExpressionFactory
IDSSExpressionFactory::NewExpression
IDSSExpressionFactory2
IDSSExpressionFactory2::NewExpression2
IDSSExtendedType
IDSSExtendedType::_NewEnum
IDSSExtendedType::Add
IDSSExtendedType::Clear
IDSSExtendedType::Count
IDSSExtendedType::HasDimty
IDSSExtendedType::IsGenerated
IDSSExtendedType::Item
IDSSExtendedType::Refresh
IDSSExtendedType::Remove

IDSSExtension
IDSSExtension::Description
IDSSExtension::ExtensionUnits
IDSSExtension::LowestAnalysisLevel
IDSSExtension::Name
IDSSExtensions
IDSSExtensions::_NewEnum
IDSSExtensions::Add
IDSSExtensions::Clear
IDSSExtensions::Count
IDSSExtensions::Item
IDSSExtensions::Remove
IDSSExtensionUnit
IDSSExtensionUnit::Attributes
IDSSExtensionUnit::Relationship
IDSSExtensionUnits
IDSSExtensionUnits::_NewEnum
IDSSExtensionUnits::Add
IDSSExtensionUnits::Clear
IDSSExtensionUnits::Count
IDSSExtensionUnits::Item
IDSSExtensionUnits::Remove
IDSSFact
IDSSFact::Column
IDSSFact::EntryLevel
IDSSFact::Extensions
IDSSFact::Info
IDSSFact::OuterJoinType
IDSSFact::Populate
IDSSFact::Tables
IDSSFactGroup
IDSSFactGroup::_NewEnum
IDSSFactGroup::Add
IDSSFactGroup::Clear
IDSSFactGroup::Count
IDSSFactGroup::Extensions
IDSSFactGroup::Info
IDSSFactGroup::Item
IDSSFactGroup::Populate
IDSSFactGroup::Remove
IDSSFactGroup2
IDSSFactGroup2::ItemByMetric
IDSSFactGroups
IDSSFactGroups::_NewEnum
IDSSFactGroups::Add
IDSSFactGroups::Clear
IDSSFactGroups::Count
IDSSFactGroups::Item
IDSSFactGroups::Remove
IDSSFactInfo
IDSSFactInfo::Expression
IDSSFactInfo::Extensions
IDSSFactInfo::Fact
IDSSFactInfo2
IDSSFactInfo2::Metric
IDSSFacts
IDSSFacts::_NewEnum
IDSSFacts::Add
IDSSFacts::Clear
IDSSFacts::Count
IDSSFacts::Item
IDSSFacts::Remove
IDSSFieldGroupIterator
IDSSFieldGroupIterator::CountFields
IDSSFieldGroupIterator::CountRows
IDSSFieldGroupIterator::Field
IDSSFieldGroupIterator::FirstRow
IDSSFieldGroupIterator::NextRow
IDSSFieldGroupIterator::Text
IDSSFieldGroupIterator2
IDSSFieldGroupIterator2::GetValue
IDSSFieldGroupIterator2::LinkText
IDSSFieldGroupIterator3
IDSSFieldGroupIterator3::GetFormatAndText
IDSSFieldGroupIterator3::IsLinkStatic
IDSSFieldGroupIterator3::Threshold
IDSSFilter
IDSSFilter::Expression
IDSSFilter::FilterType
IDSSFilter::Info
IDSSFilter::Populate
IDSSFilter::Root
IDSSFindObject
IDSSFindObject::Execute
IDSSFindObject::Flags
IDSSFindObject::Info
IDSSFindObject::MaxObjects
IDSSFindObject::NamePattern
IDSSFindObject::Owner
IDSSFindObject::Recurse
IDSSFindObject::RootFolder
IDSSFindObject::Types
IDSSFolder
IDSSFolder::_NewEnum
IDSSFolder::Add
IDSSFolder::AddCopy
IDSSFolder::AddMove
IDSSFolder::Clear
IDSSFolder::Count
IDSSFolder::Filtering
IDSSFolder::FindTypedObjects
IDSSFolder::FolderType
IDSSFolder::Info
IDSSFolder::Item
IDSSFolder::NewHierarchy
IDSSFolder::Parent
IDSSFolder::Remove
IDSSFolder::Root
IDSSFolder::Search
IDSSFolder::Special
IDSSFolder::SubfoldersCount
IDSSFolder::Types
IDSSFolder::View
IDSSFormat
IDSSFormat::Info
IDSSFormat::Populate
IDSSFormShortcut
IDSSFormShortcut::Attribute
IDSSFormShortcut::Form
IDSSFormShortcut::Node
IDSSFormShortcut::Populate
IDSSFunction
IDSSFunction::_NewEnum
IDSSFunction::Add
IDSSFunction::AnalyticalFunctionType
IDSSFunction::Clear
IDSSFunction::Count
IDSSFunction::FunctionIndex
IDSSFunction::FunctionPropertySet
IDSSFunction::FunctionType
IDSSFunction::Info
IDSSFunction::InParameterCount
IDSSFunction::IsDBOnly
IDSSFunction::IsOrderSignificant
IDSSFunction::IsPrefix
IDSSFunction::Item
IDSSFunction::OutParameterCount
IDSSFunction::Package
IDSSFunction::Parameters
IDSSFunction::PatternType
IDSSFunction::Populate
IDSSFunction::Precedence
IDSSFunction::Remove
IDSSFunction::RepeatedParameter
IDSSFunction::SQLPattern
IDSSFunction::SQLPatternByDatabase
IDSSFunction::SQLSubType
IDSSFunction::SQLType
IDSSFunction2
IDSSFunction2::PartitionFunction1
IDSSFunction2::PartitionFunction2
IDSSFunction2::RemovePattern
IDSSFunctionPackageDefinition
IDSSFunctionPackageDefinition::ClassID
IDSSFunctionPackageDefinition::Functions
IDSSFunctionPackageDefinition::Info
IDSSFunctionPackageDefinition::PackageVersion
IDSSFunctionPackageDefinition::ProgID
IDSSFunctionParameters
IDSSFunctionParameters::ParamDescription
IDSSFunctionParameters::ParameterType
IDSSFunctionParameters::ParamName
IDSSFunctions
IDSSFunctions::_NewEnum
IDSSFunctions::Add
IDSSFunctions::Clear
IDSSFunctions::Count
IDSSFunctions::Item
IDSSFunctions::Remove
IDSSGeneralCacheInfo
IDSSGeneralCacheInfo::CacheCheckCount
IDSSGeneralCacheInfo::CacheHitCount
IDSSGeneralCacheInfo::
CacheTableBucketCount
IDSSGeneralCacheInfo::
CacheTableBucketPerLock
IDSSGeneralCacheInfo::CacheUpdateCount
IDSSGeneralCacheInfo::FileCachePath
IDSSGeneralCacheInfo::
HistoricCacheCheckCount
IDSSGeneralCacheInfo::HistoricCacheHitCount
IDSSGeneralCacheInfo::
HistoricCacheUpdateCount
IDSSGeneralCacheInfo::MaxCacheCount
IDSSGeneralCacheInfo::
MaxMemoryConsumption
IDSSGeneralCacheInfo::SwapPolicy
IDSSGeneralCacheInfo::TotalCacheCount
IDSSGeneralCacheInfo::
TotalMemoryConsumption
IDSSGridFormatter
IDSSGridFormatter::BandingOption
IDSSGridFormatter::FormatCount
IDSSGridFormatter::FormatItem
IDSSGridFormatter::GetColumnScenario
IDSSGridFormatter::GetDataColumnWidths
IDSSGridFormatter::
GetDataColumnWidthsCount
IDSSGridFormatter::GetDataFormat
IDSSGridFormatter::GetDefaultHeaderWidth
IDSSGridFormatter::GetExtraWidth
IDSSGridFormatter::GetHeaderFormat
IDSSGridFormatter::GetHeaderFormat2
IDSSGridFormatter::GetRowHeaderWidths
IDSSGridFormatter::
GetRowHeaderWidthsCount
IDSSGridFormatter::GetRowHeight
IDSSGridFormatter::GetRowScenario
IDSSGridFormatter::GetTitleFormat
IDSSGridFormatter::GetTitleFormat2
IDSSGridFormatter::InitSlicedData
IDSSGridFormatter::ReportInstance
IDSSGridFormatter::Template
IDSSGridFormatter2
IDSSGridFormatter2::IsSubtotalCount
IDSSGridFormatter2::MergeColumnHeader

IDSSGridFormatter2::MergeRowHeader
IDSSGridFormatter2::ShowColumnHeader
IDSSGridFormatter2::ShowRowHeader
IDSSGridGraphIterator
IDSSGridGraphIterator::DisplayMode
IDSSGridGraphIterator::GetWindow
IDSSGridGraphIterator::ReportInstance
IDSSGridGraphIterator::SlicedGridWindow
IDSSGridGraphIterator::Template
IDSSGridGraphIterator::WindowSize
IDSSGroupByElement
IDSSGroupByElement::GroupByUnit
IDSSGroupByElement::IsCurrentElement
IDSSGroupByElement::NodeKey
IDSSGroupByElement2
IDSSGroupByElement2::DisplayText
IDSSGroupByElements
IDSSGroupByElements::CurrentElement
IDSSGroupByElements::NextCurrentElement
IDSSGroupByElements::NodeKey
IDSSGroupByElements::PreviousCurrentElement
IDSSGroupByUnit
IDSSGroupByUnit::AttributeForms
IDSSGroupByUnit::Format
IDSSGroupByUnit::GroupByObject
IDSSGroupByUnit::Index
IDSSGroupByUnit::PageThroughElement
IDSSGroupByUnit::PageThroughElementID
IDSSGroupByUnit::PageThroughElementStatus
IDSSGroupByUnit2
IDSSGroupByUnit2::CurrentElement
IDSSGroupByUnit2::CurrentElementID
IDSSGroupByUnit2::CurrentElementStatus
IDSSGroupByUnits
IDSSGroupByUnits::_NewEnum
IDSSGroupByUnits::Add
IDSSGroupByUnits::Clear
IDSSGroupByUnits::Count
IDSSGroupByUnits::Item
IDSSGroupByUnits::Remove
IDSSGroupByUnits::Sorts
IDSSImportServer
IDSSImportServer::Accept
IDSSImportServer::Cancel
IDSSImportServer::Catalog
IDSSImportServer::ChangeColumnIndex
IDSSImportServer::ChangeDBRole
IDSSImportServer::ChangedObjects
IDSSImportServer::DestinationFolder
IDSSImportServer::MapAttribute
IDSSImportServer::MapAttributeForm
IDSSImportServer::MapColumn
IDSSImportServer::MapColumns
IDSSImportServer::MapDBTable
IDSSImportServer::MapDBTables
IDSSImportServer::MapDimension
IDSSImportServer::MapFact
IDSSImportServer::MapMetric
IDSSImportServer::MapObject
IDSSImportServer::MapObjects
IDSSImportServer::MapReport
IDSSImportServer::MapTable
IDSSImportServer::MapTables
IDSSImportServer::MissingObjects
IDSSImportServer::NewObjects
IDSSImportServer::RemoveMapping
IDSSImportServer::ReplaceObject
IDSSImportServer::Schema
IDSSImportServer::XDAType
IDSSImportServer2
IDSSImportServer2::MapPrompt
IDSSImportServer2::RemoveMappingEx
IDSSImportServices
IDSSImportServices::NewImportServer
IDSSInBox
IDSSInBox::_NewEnum
IDSSInBox::Add
IDSSInBox::ChangeItemStatus
IDSSInBox::Clear
IDSSInBox::Count
IDSSInBox::GetMsgResult
IDSSInBox::IsFull
IDSSInBox::Item
IDSSInBox::MaxCount
IDSSInBox::Remove
IDSSInBox::SendMsg
IDSSInBoxMsg
IDSSInBoxMsg::ClientType
IDSSInBoxMsg::FinishTime
IDSSInBoxMsg::FolderID
IDSSInBoxMsg::FolderName
IDSSInBoxMsg::Locale
IDSSInBoxMsg::MsgCreationTime
IDSSInBoxMsg::MsgID
IDSSInBoxMsg::MsgModificationTime
IDSSInBoxMsg::MsgStatus
IDSSInBoxMsg::MsgText
IDSSInBoxMsg::MsgTitle
IDSSInBoxMsg::MsgType
IDSSInBoxMsg::ObjCreatorName
IDSSInBoxMsg::ObjDescription
IDSSInBoxMsg::ObjectID
IDSSInBoxMsg::ParentMsgID
IDSSInBoxMsg::ProjectID
IDSSInBoxMsg::ProjectName
IDSSInBoxMsg::RqstStatus
IDSSInBoxMsg::RqstType
IDSSInBoxMsg::StartTime
IDSSInBoxMsg::UserProperty
IDSSJob
IDSSJob::Close
IDSSJob::CreationTime
IDSSJob::CurrentUnit
IDSSJob::Description
IDSSJob::ErrorMessage
IDSSJob::Filter
IDSSJob::FinishedStep
IDSSJob::FinishedStepCount
IDSSJob::FinishedStepElapseTime
IDSSJob::FinishedStepFinishTime
IDSSJob::FinishedStepId
IDSSJob::FinishedStepPU
IDSSJob::FinishedStepStartTime
IDSSJob::FinishedStepThread
IDSSJob::FinishTime
IDSSJob::ID
IDSSJob::JobStatus
IDSSJob::JobType
IDSSJob::MachineName
IDSSJob::NetAddress
IDSSJob::Owner
IDSSJob::Priority
IDSSJob::ProjectId
IDSSJob::ProjectName
IDSSJob::PUCount
IDSSJob::PUEnterTime
IDSSJob::PUExitTime
IDSSJob::PUId
IDSSJob::PUPriority
IDSSJob::PUStartTime
IDSSJob::Refresh
IDSSJob::RunningThread
IDSSJob::Source
IDSSJob::SQL
IDSSJob::Status
IDSSJob::StatusTime
IDSSJob::Template
IDSSJob::TodoStep
IDSSJob::TodoStepCount
IDSSJob::TodoStepId
IDSSJob::TodoStepList
IDSSJob::TodoStepPU
IDSSJob::TodoStepStartTime
IDSSJob::TodoStepThread
IDSSJob::TotalJobPasses
IDSSJob::TotalTables
IDSSJob::UserName
IDSSJobManager
IDSSJobManager::Close
IDSSJobManager::GetJobs
IDSSJobManager::GetUserJobs
IDSSJobManager::RefreshJob
IDSSJobPriorityMap
IDSSJobPriorityMap::Add
IDSSJobPriorityMap::Count
IDSSJobPriorityMap::DefaultPriority
IDSSJobPriorityMap::Delete
IDSSJobPriorityMap::DeleteByName
IDSSJobPriorityMap::Description
IDSSJobPriorityMap::Item
IDSSJobPriorityMap::ItemByName
IDSSJobPriorityMap::JobPriorityMapEntrys
IDSSJobPriorityMap::Name
IDSSJobPriorityMap::ReportCostThreshold
IDSSJobPriorityMap::ReportCostThresholds
IDSSJobPriorityMap::ReportCostWeight
IDSSJobPriorityMapEntry
IDSSJobPriorityMapEntry::Application
IDSSJobPriorityMapEntry::Name
IDSSJobPriorityMapEntry::Priority
IDSSJobPriorityMapEntry::Project
IDSSJobPriorityMapEntry::ReportCost
IDSSJobPriorityMapEntry::ReportCostLowerLimit
IDSSJobPriorityMapEntry::ReportCostUpperLimit
IDSSJobPriorityMapEntry::RequestType
IDSSJobPriorityMapEntry::User
IDSSJobPriorityMapEntrys
IDSSJobPriorityMapEntrys::_NewEnum
IDSSJobPriorityMapEntrys::Add
IDSSJobPriorityMapEntrys::Clear
IDSSJobPriorityMapEntrys::Count
IDSSJobPriorityMapEntrys::Item
IDSSJobPriorityMapEntrys::Remove
IDSSJobs
IDSSJobs::_NewEnum
IDSSJobs::Count
IDSSJobs::Item
IDSSLDAPSettings
IDSSLDAPSettings::LDAPAttrMSTRImport
IDSSLDAPSettings::LDAPAttrMSTRUserGUID
IDSSLDAPSettings::LDAPDLLs
IDSSLDAPSettings::LDAPImportGroup
IDSSLDAPSettings::LDAPImportUser
IDSSLDAPSettings::LDAPMSTRAuthUserDN
IDSSLDAPSettings::LDAPMSTRAuthUserPwd
IDSSLDAPSettings::LDAPSearchRoot
IDSSLDAPSettings::LDAPServerCertificate
IDSSLDAPSettings::LDAPServerName
IDSSLDAPSettings::LDAPServerPort
IDSSLDAPSettings::LDAPServerSecure Connection
IDSSLDAPSettings::LDAPTimeOut
IDSSLDAPSettings::LDAPVendor
IDSSLDAPSettings::LDAPVersion
IDSSLDAPSettings::LoadFromXMLDOMStr
IDSSLDAPSettings::SaveToXMLDOMStr

IDSSLDAPSettings2
IDSSLDAPSettings2::LDAPAttributeForLogin
IDSSLDAPSettings2::LDAPObjectClassFor UserGroup
IDSSLDAPSettings3
IDSSLDAPSettings3::LDAPAttributeFor GroupName
IDSSLDAPSettings3:: LDAPAttributeForUserName
IDSSLDAPSettings3::LDAPSearchFilterForDN FromLogin
IDSSLDAPSettings3:: LDAPSearchFilterForGroup
IDSSLDAPSettings3::LDAPStringCoding
IDSSLDAPSettings3::LDAPSynch
IDSSLDAPSettings4
IDSSLDAPSettings4::CallConvention
IDSSLDAPSettings4::LDAPServerVendor
IDSSLDAPSettings4::Platform
IDSSLicenseComplianceManager
IDSSLicenseComplianceManager:: CheckCompliance
IDSSLicenseComplianceManager:: CheckSpecificCompliance
IDSSLicenseComplianceManager:: GetCPUComplianceDetails
IDSSLicenseComplianceManager::GetNamed UserComplianceDetails
IDSSLicenseComplianceManager2
IDSSLicenseComplianceManager2:: GetComplianceLevel
IDSSLicenseComplianceManager2:: GetCPUOutofComplianceDetails
IDSSLicenseComplianceManager2::GetNamed UserOutofComplianceDetails
IDSSLicenseComplianceManager2:: PerformComplianceCheck
IDSSLicenseTypes
IDSSLicenseTypes::Count
IDSSLicenseTypes::Group
IDSSLicenseTypes::LicenseType
IDSSLicenseTypes::Privilege
IDSSLicenseTypes::Project
IDSSLicenseTypes::SecurityRole
IDSSLicenseTypes::Trustee
IDSSLink
IDSSLink::_NewEnum
IDSSLink::ClearAllItems
IDSSLink::DefaultFromLink
IDSSLink::DefaultFromType
IDSSLink::Info
IDSSLink::IsModified
IDSSLink::LinkPropertySets
IDSSLink::NewLinkItems
IDSSLink::Populate
IDSSLink::SaveAllItems
IDSSLink::Types
IDSSLink2
IDSSLink2::PurgeCache
IDSSLinkItem
IDSSLinkItem::_NewEnum
IDSSLinkItem::Clear
IDSSLinkItem::Count
IDSSLinkItem::Delete
IDSSLinkItem::IsModified
IDSSLinkItem::Item
IDSSLinkItem::ItemByType
IDSSLinkItem::Populate
IDSSLinkItem::PropertySets
IDSSLinkItem::Refresh
IDSSLinkItem::Save
IDSSLinkItems
IDSSLinkItems::_NewEnum
IDSSLinkItems::AddCopy
IDSSLinkItems::AddItem
IDSSLinkItems::Clear
IDSSLinkItems::Complete
IDSSLinkItems::Count
IDSSLinkItems::DeleteItems
IDSSLinkItems::ExcludeTrivialItems
IDSSLinkItems::FindItems
IDSSLinkItems::FindItemsEx
IDSSLinkItems::FindItemsWithDefaults
IDSSLinkItems::IsModified
IDSSLinkItems::Item
IDSSLinkItems::MaxCount
IDSSLinkItems::Offset
IDSSLinkItems::SaveItems
IDSSLinkItems::SaveModifiedItems
IDSSLinkItems::Total
IDSSLinkItems::Uses
IDSSMasterEnum
IDSSMDMover
IDSSMDMover::Move
IDSSMDSecurityFilter
IDSSMDSecurityFilter::BottomLevel
IDSSMDSecurityFilter::Expression
IDSSMDSecurityFilter::Info
IDSSMDSecurityFilter::Populate
IDSSMDSecurityFilter::Root
IDSSMDSecurityFilter::TopLevel
IDSSMDUpdate
IDSSMDUpdate::CurrentVersion
IDSSMDUpdate::NeedUpdate
IDSSMDUpdate::Update
IDSSMDUpdate2
IDSSMDUpdate2::UpdateSecurityModel
IDSSMDUpdate3
IDSSMDUpdate3::UpdateDocumentDefinitions
IDSSMDXColumn
IDSSMDXColumn::MDXDimensionUniqueName
IDSSMDXColumn::MDXHierarchyUniqueName
IDSSMDXColumn::MDXLevelNumber
IDSSMDXColumn::MDXObjectName
IDSSMDXColumn::MDXObjectProperty
IDSSMDXColumn::MDXObjectType
IDSSMDXColumn::MDXObjectUniqueName
IDSSMDXColumn::XDAColumn
IDSSMDXCube
IDSSMDXCube::CatalogDescription
IDSSMDXCube::CatalogName
IDSSMDXCube::CubeName
IDSSMDXCube::CubeType
IDSSMDXCube::Dimensions
IDSSMDXCube::LastDataUpdate
IDSSMDXCube::LastSchemaUpdate
IDSSMDXCube::RefreshStatus
IDSSMDXCube::SchemaName
IDSSMDXCube::SharedObjects
IDSSMDXCube::XDADBTable
IDSSMDXCube2
IDSSMDXCube2::ImpliedDimensions
IDSSMDXCube2::SAPVariables
IDSSMDXDimension
IDSSMDXDimension::DefaultHierarchy
IDSSMDXDimension::Description
IDSSMDXDimension::DimensionCaption
IDSSMDXDimension::DimensionName
IDSSMDXDimension::DimensionType
IDSSMDXDimension::DimensionUniqueName
IDSSMDXDimension::Hierarchies
IDSSMDXDimension::IsShared
IDSSMDXDimension::Populate
IDSSMDXDimension::RefreshStatus
IDSSMDXDimensions
IDSSMDXDimensions::_NewEnum
IDSSMDXDimensions::Add
IDSSMDXDimensions::Clear
IDSSMDXDimensions::Count
IDSSMDXDimensions::Item
IDSSMDXDimensions::ItemNoRelaying
IDSSMDXDimensions::Remove
IDSSMDXHierarchies
IDSSMDXHierarchies::_NewEnum
IDSSMDXHierarchies::Add
IDSSMDXHierarchies::Clear
IDSSMDXHierarchies::Count
IDSSMDXHierarchies::Item
IDSSMDXHierarchies::Remove
IDSSMDXHierarchy
IDSSMDXHierarchy::AllMember
IDSSMDXHierarchy::DefaultMember
IDSSMDXHierarchy::Description
IDSSMDXHierarchy::DimensionUniqueName
IDSSMDXHierarchy::HierarchyCaption
IDSSMDXHierarchy::HierarchyName
IDSSMDXHierarchy::HierarchyUniqueName
IDSSMDXHierarchy::Levels
IDSSMDXHierarchy::RefreshStatus
IDSSMDXHierarchy::Structure
IDSSMDXProperty
IDSSMDXSAPVariable
IDSSMDXSAPVariable::CharacterMaxLength
IDSSMDXSAPVariable::DataType
IDSSMDXSAPVariable::DefaultHigh
IDSSMDXSAPVariable::DefaultHighCaption
IDSSMDXSAPVariable::DefaultLow
IDSSMDXSAPVariable::DefaultLowCaption
IDSSMDXSAPVariable::Description
IDSSMDXSAPVariable::Populate
IDSSMDXSAPVariable::ReferenceDimension
IDSSMDXSAPVariable::ReferenceHierarchy
IDSSMDXSAPVariable::VariableCaption
IDSSMDXSAPVariable::VariableInputType
IDSSMDXSAPVariable::VariableName
IDSSMDXSAPVariable::VariableOrdinal
IDSSMDXSAPVariable::VariableProcessingType
IDSSMDXSAPVariable::VariableSelectionType
IDSSMDXSAPVariable::VariableType
IDSSMDXSAPVariable::VariableUID
IDSSMDXSAPVariableInfo
IDSSMDXSAPVariableInfo::Answer
IDSSMDXSAPVariableInfo::IsKeyDate
IDSSMDXSAPVariableInfo::Prompt
IDSSMDXSAPVariableInfo::PromptType
IDSSMDXSAPVariableInfo::VariableName
IDSSMDXSAPVariableInfos
IDSSMDXSAPVariableInfos::_NewEnum
IDSSMDXSAPVariableInfos::Add
IDSSMDXSAPVariableInfos::Clear
IDSSMDXSAPVariableInfos::Count
IDSSMDXSAPVariableInfos::Item
IDSSMDXSAPVariableInfos::Remove
IDSSMDXSAPVariables
IDSSMDXSAPVariables::_NewEnum
IDSSMDXSAPVariables::Add
IDSSMDXSAPVariables::Clear
IDSSMDXSAPVariables::Count
IDSSMDXSAPVariables::Item
IDSSMDXSAPVariables::Remove
IDSSMDXSharedObjects
IDSSMDXSharedObjects::Dimensions
IDSSMDXSharedObjects::XDACatalog
IDSSMDXSharedObjects2
IDSSMDXSharedObjects2::SAPVariables
IDSSMetadataServer

IDSSMetadataServer::ChangeObjectID
IDSSMetadataServer::ContentSource
IDSSMetadataServer::ListConfigs
IDSSMetadataServer::ListProjects
IDSSMetadataServer::PersistObjectIDs
IDSSMetadataServer::SystemUser
IDSSMetadataServer2
IDSSMetadataServer2::RealSystemUser
IDSSMetric
IDSSMetric::AllChildrenJoin
IDSSMetric::Column
IDSSMetric::Conditionality
IDSSMetric::Dimensionality
IDSSMetric::DimensionalityPrompt
IDSSMetric::Expression
IDSSMetric::Info
IDSSMetric::IsDecomposable
IDSSMetric::MetricType
IDSSMetric::Populate
IDSSMetric::Root
IDSSMetric::Subtotals
IDSSMetric2
IDSSMetric2::DataType
IDSSMetric2::GridFormat
IDSSMetric2::HeaderFormat
IDSSMetric2::MetricText
IDSSMetric3
IDSSMetric3::AggregateFromBase
IDSSMetricInfo
IDSSMetricInfo::Expression
IDSSMetricInfo::Metric
IDSSMetricInfos
IDSSMetricInfos::_NewEnum
IDSSMetricInfos::Add
IDSSMetricInfos::Clear
IDSSMetricInfos::Count
IDSSMetricInfos::Item
IDSSMetricInfos::Remove
IDSSMetrics
IDSSMetrics::_NewEnum
IDSSMetrics::Add
IDSSMetrics::Clear
IDSSMetrics::Count
IDSSMetrics::Item
IDSSMetrics::Remove
IDSSMetricSubtotal
IDSSMetricSubtotal::Default
IDSSMetricSubtotal::Definition
IDSSMetricSubtotal::Implementation
IDSSMetricSubtotal::Metric
IDSSMetricSubtotal::Name
IDSSMetricSubtotals
IDSSMetricSubtotals::_NewEnum
IDSSMetricSubtotals::Add
IDSSMetricSubtotals::Clear
IDSSMetricSubtotals::Count
IDSSMetricSubtotals::Item
IDSSMetricSubtotals::Remove
IDSSMetricSubtotals::SetDefault
IDSSMonitor
IDSSMonitor::_NewEnum
IDSSMonitor::Add
IDSSMonitor::Clear
IDSSMonitor::Count
IDSSMonitor::Info
IDSSMonitor::Item
IDSSMonitor::MonitorType
IDSSMonitor::Populate
IDSSMonitor::Remove
IDSSMonitorField
IDSSMonitorField::Clear
IDSSMonitorField::Name
IDSSMonitorField::Property
IDSSMsg
IDSSMsg::AutoCommit
IDSSMsg::JobID
IDSSMsg::MsgBegin
IDSSMsg::PackBinary
IDSSMsg::PackBSTR
IDSSMsg::PackDouble
IDSSMsg::PackInt
IDSSMsg::PackLong
IDSSMsg::Reset
IDSSMsg::SessionID
IDSSMsg::UnPackBinary
IDSSMsg::UnPackBinarySize
IDSSMsg::UnPackBSTR
IDSSMsg::UnPackDouble
IDSSMsg::UnPackInt
IDSSMsg::UnPackLong
IDSSMsgProcessor
IDSSMsgProcessor::Process
IDSSMX2ETLMapping
IDSSMX2ETLMapping::Name
IDSSMX2ETLMapping::SessionLog
IDSSMX2ETLMapping::Sources
IDSSMX2ETLMapping::Targets
IDSSMX2ETLMappings
IDSSMX2ETLMappings::_NewEnum
IDSSMX2ETLMappings::Count
IDSSMX2ETLMappings::Item
IDSSMX2ETLServer
IDSSMX2ETLServer::Clear
IDSSMX2ETLServer::GetObjectMappings
IDSSMX2ETLServer::GetTableMappings
IDSSMX2ETLServer::IsMX2Available
IDSSMX2ETLServer::Refresh
IDSSMX2ETLSessionLog
IDSSMX2ETLSessionLog::LogFileName
IDSSMX2ETLSessionLog::RowsFailed
IDSSMX2ETLSessionLog::RowsPassed
IDSSMX2ETLSessionLog::SessionName
IDSSMX2ETLSessionLog::TimeDone
IDSSMX2ETLSessionLog::TimeStarted
IDSSMX2ETLSource
IDSSMX2ETLSource::_NewEnum
IDSSMX2ETLSource::Count
IDSSMX2ETLSource::Item
IDSSMX2ETLSource::Name
IDSSMX2ETLSourceField
IDSSMX2ETLSourceField::Name
IDSSMX2ETLSources
IDSSMX2ETLSources::_NewEnum
IDSSMX2ETLSources::Count
IDSSMX2ETLSources::Item
IDSSMX2ETLTarget
IDSSMX2ETLTarget::_NewEnum
IDSSMX2ETLTarget::Count
IDSSMX2ETLTarget::Item
IDSSMX2ETLTarget::Name
IDSSMX2ETLTargetField
IDSSMX2ETLTargetField::Name
IDSSMX2ETLTargetField::TransformExpression
IDSSMX2ETLTargets
IDSSMX2ETLTargets::_NewEnum
IDSSMX2ETLTargets::Count
IDSSMX2ETLTargets::Item
IDSSNode
IDSSNode::_NewEnum
IDSSNode::Add
IDSSNode::AddCopy
IDSSNode::AddMove
IDSSNode::CanExpand
IDSSNode::Child
IDSSNode::ChildIndex
IDSSNode::Clear
IDSSNode::Count
IDSSNode::DataType
IDSSNode::Dimensionality
IDSSNode::DimensionalityPrompt
IDSSNode::DimensionalityType
IDSSNode::ExpressionType
IDSSNode::FirstSibling
IDSSNode::IsSpliced
IDSSNode::Item
IDSSNode::Key
IDSSNode::LastSibling
IDSSNode::Name
IDSSNode::Next
IDSSNode::Parent
IDSSNode::Previous
IDSSNode::Remove
IDSSNode::Root
IDSSNode::Special
IDSSNode::Tag
IDSSNode::Text
IDSSNode::TextEx
IDSSNode::Type
IDSSNode2
IDSSNode2::IsIndependent
IDSSNode3
IDSSNode3::ChildGridFormat
IDSSNode3::ChildHeaderFormat
IDSSNode3::GridFormat
IDSSNode3::HeaderFormat
IDSSNode4
IDSSNode4::PopulateChild
IDSSObjectInfo
IDSSObjectInfo::Abbreviation
IDSSObjectInfo::AccessGranted
IDSSObjectInfo::AccessGrantedStr
IDSSObjectInfo::Comments
IDSSObjectInfo::Copy
IDSSObjectInfo::CreationTime
IDSSObjectInfo::Delete
IDSSObjectInfo::Dependents
IDSSObjectInfo::Description
IDSSObjectInfo::Embedded
IDSSObjectInfo::Errors
IDSSObjectInfo::Hidden
IDSSObjectInfo::ID
IDSSObjectInfo::InitialObject
IDSSObjectInfo::Load
IDSSObjectInfo::LoadFromXMLDOM
IDSSObjectInfo::ModificationTime
IDSSObjectInfo::Name
IDSSObjectInfo::Owner
IDSSObjectInfo::Parent
IDSSObjectInfo::Path
IDSSObjectInfo::PopulateObject
IDSSObjectInfo::Prompts
IDSSObjectInfo::PropertySets
IDSSObjectInfo::Refresh
IDSSObjectInfo::Save
IDSSObjectInfo::SaveToXMLDOM
IDSSObjectInfo::Security
IDSSObjectInfo::Source
IDSSObjectInfo::Special
IDSSObjectInfo::State
IDSSObjectInfo::SubType
IDSSObjectInfo::Tag
IDSSObjectInfo::Type
IDSSObjectInfo::Validate
IDSSObjectInfo::Version
IDSSObjectInfo2
IDSSObjectInfo2::DestinationFolder

IDSSObjectInfo3
IDSSObjectInfo3::EffectiveVersionRequired ForAccess
IDSSObjectInfo3::IsManaged
IDSSObjectInfo3::VersionRequiredForAccess
IDSSObjectManipulator
IDSSObjectManipulator::DeepCopyObject
IDSSObjectManipulator::DeepDeleteObject
IDSSObjectSecurity
IDSSObjectSecurity::AccessList
IDSSObjectSecurity::AccessRightCount
IDSSObjectSecurity::AccessRightDescription
IDSSObjectSecurity::AccessRightName
IDSSObjectSecurity::AccessRightNames
IDSSObjectSecurity::PopulateACL
IDSSObjectSecurity::RefreshAccess
IDSSObjectSecurity::TakeOwnership
IDSSObjectSecurity::UseDefault
IDSSObjectServer
IDSSObjectServer::Source
IDSSOLAPSorts
IDSSOLAPSorts::_NewEnum
IDSSOLAPSorts::Add
IDSSOLAPSorts::Clear
IDSSOLAPSorts::Count
IDSSOLAPSorts::Item
IDSSOLAPSorts::Remove
IDSSOLAPSorts::Sorts
IDSSOperator
IDSSOperator::Function
IDSSOperator::FunctionType
IDSSOperator::Node
IDSSOperator::OLAPSorts
IDSSOperator::Prompt
IDSSOperator::Properties
IDSSParameterPattern
IDSSParameterPattern::DataType
IDSSParameterPattern::InParameterCount
IDSSParameterPattern::Name
IDSSParameterPattern::ObjectType
IDSSParameterPattern::OutParameterCount
IDSSParameterPattern::PatternIndex
IDSSParameterPattern::RepeatedParameter
IDSSParserXgi
IDSSParserXgi::ExpressionNode
IDSSParserXgi::Flags
IDSSParserXgi::InitParsing
IDSSParserXgi::Locale
IDSSParserXgi::LocalSymbolFolder
IDSSParserXgi::NextToken
IDSSParserXgi::Parse
IDSSParserXgi::ParseTokenStream
IDSSParserXgi::SearchObject
IDSSParserXgi::SearchResultsFolder
IDSSParserXgi::SourceObject
IDSSParserXgi::TargetObject
IDSSParserXgi::TargetType
IDSSParserXgi::TokenizeObject
IDSSParserXgi::TokenStream
IDSSParserXgi2
IDSSParserXgi2::ExtraNameSpace
IDSSParserXgi3
IDSSParserXgi3::AddDocumentNameSpace
IDSSParserXgi3::ClearDocumentNameSpace
IDSSPrivileges
IDSSPrivileges::_NewEnum
IDSSPrivileges::Add
IDSSPrivileges::BitMask
IDSSPrivileges::Clear
IDSSPrivileges::Count
IDSSPrivileges::Description
IDSSPrivileges::Includes
IDSSPrivileges::Item
IDSSPrivileges::Name
IDSSPrivileges::Remove
IDSSPrivileges::SysCount
IDSSProcessingUnit
IDSSProcessingUnit::Name
IDSSProcessingUnit::Queue
IDSSProcessingUnit::Setting
IDSSProcessingUnit::Tasks
IDSSProcessingUnits
IDSSProcessingUnits::_NewEnum
IDSSProcessingUnits::Add
IDSSProcessingUnits::Clear
IDSSProcessingUnits::Count
IDSSProcessingUnits::Item
IDSSProcessingUnits::Remove
IDSSProject
IDSSProject::Catalog
IDSSProject::Configuration
IDSSProject::DBConnectionMaps
IDSSProject::DrillMap
IDSSProject::Info
IDSSProject::LoginUser
IDSSProject::LogoutUser
IDSSProject::Populate
IDSSProject::RoleMemberships
IDSSProject::Root
IDSSProject::Schema
IDSSProject::SystemFunction
IDSSProject::SystemLink
IDSSProject::SystemPropertySet
IDSSProject::SystemSubtotal
IDSSProject::UnitDrillMap
IDSSProject::WarehouseDBRoles
IDSSProject2
IDSSProject2::SecurityModel
IDSSProject3
IDSSProject3::ManagedObjects
IDSSProject3::SystemPrompts
IDSSProject4
IDSSProject4::AllWHExecutionDBRoles
IDSSProject4::WHAuthenticationDBRole
IDSSProject4::WHExecutionDBRoles
IDSSProjectInstance
IDSSProjectInstance::DBConnection
IDSSProjectInstance::DBLogin
IDSSProjectInstance::ID
IDSSProjectInstance::Idle
IDSSProjectInstance::Load
IDSSProjectInstance::MetadataVersion
IDSSProjectInstance::Name
IDSSProjectInstance::ProjectDescription
IDSSProjectInstance::ProjectID
IDSSProjectInstance::ProjectName
IDSSProjectInstance::ProjectReferenceName
IDSSProjectInstance::Resume
IDSSProjectInstance::Status
IDSSProjectInstance::Unload
IDSSProjectInstances
IDSSProjectInstances::_NewEnum
IDSSProjectInstances::Count
IDSSProjectInstances::Item
IDSSProjectInstances::ItemByIndex
IDSSProjectInstances::ItemByName
IDSSProjectReference
IDSSProjectReference::DBConnection
IDSSProjectReference::DBLogin
IDSSProjectReference::FileName
IDSSProjectReference::ID
IDSSProjectReference::InternalProjectID
IDSSProjectReference::LoadFlag
IDSSProjectReference::MetadataDBRole
IDSSProjectReference::MetadataVersion
IDSSProjectReference::Name
IDSSProjectReference::Setting
IDSSProjectReference::SettingCount
IDSSProjectReference::StatisticsDBConnection
IDSSProjectReference::StatisticsDBLogin
IDSSProjectReference::StatisticsDBRole
IDSSProjectReference::StatisticSetting
IDSSProjectReference:: WarehouseDBConnection
IDSSProjectReference::WarehouseDBLogin
IDSSProjectReference2
IDSSProjectReference2::Populate
IDSSProjectReference3
IDSSProjectReference3::StatisticTableSetting
IDSSProjectReferences
IDSSProjectReferences::_NewEnum
IDSSProjectReferences::Add
IDSSProjectReferences::Clear
IDSSProjectReferences::Count
IDSSProjectReferences::Item
IDSSProjectReferences::Remove
IDSSPrompt
IDSSPrompt::Answer
IDSSPrompt::AnswerList
IDSSPrompt::Auxiliary
IDSSPrompt::AuxiliaryExports
IDSSPrompt::AuxiliaryPrompt
IDSSPrompt::Closed
IDSSPrompt::Default
IDSSPrompt::ExportsToPrompt
IDSSPrompt::HasPrevious
IDSSPrompt::Importable
IDSSPrompt::ImportAs
IDSSPrompt::Incomplete
IDSSPrompt::Index
IDSSPrompt::Info
IDSSPrompt::IsEquivalentAnswer
IDSSPrompt::Locations
IDSSPrompt::Locked
IDSSPrompt::Maximum
IDSSPrompt::MaximumExports
IDSSPrompt::MaximumPrompt
IDSSPrompt::Meaning
IDSSPrompt::Merge
IDSSPrompt::Minimum
IDSSPrompt::MinimumExports
IDSSPrompt::MinimumPrompt
IDSSPrompt::Origin
IDSSPrompt::OriginExports
IDSSPrompt::OriginPrompt
IDSSPrompt::Populate
IDSSPrompt::Previous
IDSSPrompt::Prompts
IDSSPrompt::PromptType
IDSSPrompt::Restriction
IDSSPrompt::RestrictionExports
IDSSPrompt::RestrictionPrompt
IDSSPrompt::Reuse
IDSSPrompt::Title
IDSSPrompt::Used
IDSSPrompt2
IDSSPrompt2::IsDormant
IDSSPrompt3
IDSSPrompt3::SystemPromptType
IDSSPromptEvaluator
IDSSPromptEvaluator::Evaluate
IDSSPromptInfo
IDSSPromptInfo::ReportHasPrompts
IDSSPromptInfo::ResolvedSecurityFilter
IDSSPromptInfo::SecurityFilterHasPrompts
IDSSPromptInfo::SecurityFilterPromptCount

IDSSPromptLocation
IDSSPromptLocation::Container
IDSSPromptLocation::Index
IDSSPromptLocation::Prompt
IDSSPromptLocations
IDSSPromptLocations::_NewEnum
IDSSPromptLocations::Count
IDSSPromptLocations::Item
IDSSPromptLocations2
IDSSPromptLocations2::ReportSource
IDSSPromptLocations2::ReportSourceType
IDSSPromptLocations2::SourceCount
IDSSPromptLocations2::Table
IDSSPromptLocations2::XDAType
IDSSPrompts
IDSSPrompts::_NewEnum
IDSSPrompts::Add
IDSSPrompts::Clear
IDSSPrompts::Count
IDSSPrompts::Item
IDSSPrompts::Remove
IDSSProperty
IDSSProperty::Default
IDSSProperty::Description
IDSSProperty::EnumerationCount
IDSSProperty::EnumerationDescription
IDSSProperty::EnumerationName
IDSSProperty::Hidden
IDSSProperty::Name
IDSSProperty::PropertyGroups
IDSSProperty::PropertyID
IDSSProperty::PropertyType
IDSSProperty::UseDefault
IDSSProperty::Value
IDSSPropertyGroup
IDSSPropertyGroup::_NewEnum
IDSSPropertyGroup::Add
IDSSPropertyGroup::Clear
IDSSPropertyGroup::Count
IDSSPropertyGroup::Item
IDSSPropertyGroup::Remove
IDSSPropertyHelper
IDSSPropertyHelper::LookUpPropertyDefault
IDSSPropertyHelper::LookUpPropertyLocation
IDSSPropertyRestriction
IDSSPropertyRestriction::FunctionType
IDSSPropertyRestriction::Property
IDSSPropertyRestriction::PropertySet
IDSSPropertyRestriction::Value
IDSSPropertyRestrictions
IDSSPropertyRestrictions::_NewEnum
IDSSPropertyRestrictions::Add
IDSSPropertyRestrictions::Clear
IDSSPropertyRestrictions::Count
IDSSPropertyRestrictions::Item
IDSSPropertyRestrictions::Remove
IDSSPropertySet
IDSSPropertySet::_NewEnum
IDSSPropertySet::Add
IDSSPropertySet::Clear
IDSSPropertySet::Count
IDSSPropertySet::Info
IDSSPropertySet::Item
IDSSPropertySet::Populate
IDSSPropertySet::PropertyGroups
IDSSPropertySet::Remove
IDSSQBColumn
IDSSQBColumn::Column
IDSSQBColumn::Table
IDSSQBJoin
IDSSQBJoin::Condition
IDSSQBJoin::JoinOrder
IDSSQBJoin::JoinType
IDSSQBJoin::LeftTable
IDSSQBJoin::RightTable
IDSSQBJoins
IDSSQBJoins::_NewEnum
IDSSQBJoins::Add
IDSSQBJoins::Clear
IDSSQBJoins::Count
IDSSQBJoins::Item
IDSSQBJoins::Remove
IDSSQBJoinTreeBranchNode
IDSSQBJoinTreeBranchNode::Condition
IDSSQBJoinTreeBranchNode::JoinType
IDSSQBJoinTreeBranchNode::LeftChild
IDSSQBJoinTreeBranchNode::Node
IDSSQBJoinTreeBranchNode::RightChild
IDSSQBJoinTreeLeafNode
IDSSQBJoinTreeLeafNode::QBTable
IDSSQBJoinTreeNode
IDSSQBJoinTreeNode::IsLeaf
IDSSQBJoinTreeNode::Special
IDSSQBQueryColumn
IDSSQBQueryColumn::Alias
IDSSQBQueryColumn::Expression
IDSSQBQueryColumn::GroupBy
IDSSQBQueryColumn::OrderByAscending
IDSSQBQueryColumn::OrderByPosition
IDSSQBTable
IDSSQBTable::Alias
IDSSQBTable::DbTable
IDSSQBTables
IDSSQBTables::_NewEnum
IDSSQBTables::Add
IDSSQBTables::Clear
IDSSQBTables::Count
IDSSQBTables::Item
IDSSQBTables::Remove
IDSSQueryStatColumnDef
IDSSQueryStatColumnDef::DssID
IDSSQueryStatColumnDef::Name
IDSSQueryStatistics
IDSSQueryStatistics::DisplayStringCount
IDSSQueryStatistics::GetFieldCount
IDSSQueryStatistics::GetFieldValue
IDSSQueryStatistics::GetPointerByIndex
IDSSQueryStatistics::ithDisplayString
IDSSQueryStatisticsDBData
IDSSQueryStatisticsDBData::DBInstanceID
IDSSQueryStatisticsDBData::DBInstanceName
IDSSQueryStatisticsDBData::DBUserName
IDSSQueryStatisticsPassData
IDSSQueryStatisticsPassData::GetFieldCount
IDSSQueryStatisticsPassData::GetFieldValue
IDSSQueryStatisticsPassData::GetPointerByIndex
IDSSQueryStatisticsTableData
IDSSQueryStatisticsTableData::ColumnCount
IDSSQueryStatisticsTableData::
GetColumnDataByIndex
IDSSQueryStatisticsTableData::TableID
IDSSQueryStatisticsTableData::TableName
IDSSQueryStatisticsTableData::UsageCount
IDSSQueryStatPassTableColData
IDSSQueryStatPassTableColData::
ClauseDataCount
IDSSQueryStatPassTableColData::
ColumnDefinition
IDSSQueryStatPassTableColData::
GetClauseDataByIndex
IDSSQueryStatSQLClauseData
IDSSQueryStatSQLClauseData::SQLClauseType
IDSSQueryStatSQLClauseData::UsageCount
IDSSQueue
IDSSQueue::InitThreadCount
IDSSQueue::JobPriorityMap
IDSSQueue::MaxThreadCount
IDSSQueue::MinThreadCount
IDSSQueue::Name
IDSSQueue::PriorityLowerBound
IDSSQueue::PriorityUpperBound
IDSSQueue::SubQueues
IDSSQueue2
IDSSQueue2::QueueServiceScheme
IDSSQueues
IDSSQueues::_NewEnum
IDSSQueues::Add
IDSSQueues::Clear
IDSSQueues::Count
IDSSQueues::Item
IDSSQueues::Remove
IDSSRawTabularData
IDSSRawTabularData::ColumnDataType
IDSSRawTabularData::ColumnName
IDSSRelationship
IDSSRelationship::ERType
IDSSRelationship::Expression
IDSSRelationship::Guide
IDSSRelationship::JoinAttributes
IDSSRelationship::JoinType
IDSSRelationship::PartialType
IDSSRelationship::Populate
IDSSRelationship::RelationshipType
IDSSRelationshipNode
IDSSRelationshipNode::ApplySubExpression
IDSSRelationshipNode::Node
IDSSRelationshipNode::Prompt
IDSSRelationshipNode::Relationship
IDSSRelationshipNode::UseSchema
IDSSReportCacheAdmin
IDSSReportCacheAdmin::BackupAllCaches
IDSSReportCacheAdmin::BackupCache
IDSSReportCacheAdmin::BackupCachePool
IDSSReportCacheAdmin::BackupLKUPTable
IDSSReportCacheAdmin::CacheLifeTime
IDSSReportCacheAdmin::
CacheTableBucketCount
IDSSReportCacheAdmin::
CacheTableBucketPerLock
IDSSReportCacheAdmin::CleanUpInvalidCaches
IDSSReportCacheAdmin::CreateCache
IDSSReportCacheAdmin::DeleteAllCaches
IDSSReportCacheAdmin::DeleteCache
IDSSReportCacheAdmin::ExpireAllCaches
IDSSReportCacheAdmin::ExpireCache
IDSSReportCacheAdmin::FileCachePath
IDSSReportCacheAdmin::GetAllCacheInfos
IDSSReportCacheAdmin::GetCacheInfo
IDSSReportCacheAdmin::GetCacheInfos
IDSSReportCacheAdmin::
GetClusterCacheHitCounts
IDSSReportCacheAdmin::GetGeneralCacheInfo
IDSSReportCacheAdmin::InvalidateAllCaches
IDSSReportCacheAdmin::InvalidateCache
IDSSReportCacheAdmin::LoadAllCaches
IDSSReportCacheAdmin::LoadCache
IDSSReportCacheAdmin::MaxCacheCount
IDSSReportCacheAdmin::
MaxMemoryConsumption
IDSSReportCacheAdmin::SwapPolicy
IDSSReportCacheAdmin::UnloadAllCaches
IDSSReportCacheAdmin::UnloadCache
IDSSReportCacheAdmin::UpdateAllCaches
IDSSReportCacheAdmin::UpdateCache
IDSSReportCacheInfo
IDSSReportCacheInfo::CacheFileName

IDSSReportCacheInfo::CacheID
IDSSReportCacheInfo::CacheLifeTime
IDSSReportCacheInfo::CacheSize
IDSSReportCacheInfo::CacheStatus
IDSSReportCacheInfo::CacheType
IDSSReportCacheInfo::CreateTime
IDSSReportCacheInfo::ExpireTime
IDSSReportCacheInfo::FilterID
IDSSReportCacheInfo::FilterVersionID
IDSSReportCacheInfo::GetWaitingJob
IDSSReportCacheInfo::GetWaitingJobsCount
IDSSReportCacheInfo::HistoricCheck
IDSSReportCacheInfo::HistoricHitCount
IDSSReportCacheInfo::HistoricUpdate
IDSSReportCacheInfo::InboxRefCount
IDSSReportCacheInfo::LastHitTime
IDSSReportCacheInfo::LastLoadTime
IDSSReportCacheInfo::LastUpdateTime
IDSSReportCacheInfo::LoadCheck
IDSSReportCacheInfo::LoadHitCount
IDSSReportCacheInfo::LoadUpdate
IDSSReportCacheInfo::ReportID
IDSSReportCacheInfo::ReportName
IDSSReportCacheInfo::ResultUpdateTime
IDSSReportCacheInfo::TemplateID
IDSSReportCacheInfo::TemplateVersionID
IDSSReportCacheInfo::Version
IDSSReportCacheInfos
IDSSReportCacheInfos::_NewEnum
IDSSReportCacheInfos::Add
IDSSReportCacheInfos::Clear
IDSSReportCacheInfos::Count
IDSSReportCacheInfos::Item
IDSSReportCacheInfos::Remove
IDSSReportDefinition
IDSSReportDefinition::BaseReport
IDSSReportDefinition::DeclinePrompts
IDSSReportDefinition::DisplayMode
IDSSReportDefinition::Filter
IDSSReportDefinition::FilterPrompt
IDSSReportDefinition::HistoricalCost
IDSSReportDefinition::Info
IDSSReportDefinition::Populate
IDSSReportDefinition::PropertyTargets
IDSSReportDefinition::ReportType
IDSSReportDefinition::Resolution
IDSSReportDefinition::Template
IDSSReportDefinition::TemplatePrompt
IDSSReportDefinition::UserSuppliedCost
IDSSReportDefinition::Views
IDSSReportDefinition2
IDSSReportDefinition2::DatamartReport
IDSSReportDefinition3
IDSSReportDefinition3::ViewReport
IDSSReportDefinition4
IDSSReportDefinition4::SourceTable
IDSSReportDefinition4::XDAType
IDSSReportDefinition5
IDSSReportDefinition5::DataFilter
IDSSReportDefinition5::DataTemplate
IDSSReportDefinition5::ViewFilter
IDSSReportDefinition5::ViewTemplate
IDSSReportDefinition5::ViewTemplateByIndex
IDSSReportExecutor
IDSSReportExecutor::Init
IDSSReportExecutor::ReportSource
IDSSReportExecutor::UnInit
IDSSReportInstance
IDSSReportInstance::Alerts
IDSSReportInstance::ApplyDrill
IDSSReportInstance::AppObjects
IDSSReportInstance::BaseInstance
IDSSReportInstance::BaseReportInstance
IDSSReportInstance::CancelExecution
IDSSReportInstance::ClearAll
IDSSReportInstance::Cookie
IDSSReportInstance::Definition
IDSSReportInstance::DrillAction
IDSSReportInstance::DrillElements
IDSSReportInstance::DrillPaths
IDSSReportInstance::DrillTemplateUnit
IDSSReportInstance::DrillView
IDSSReportInstance::EstimatedCost
IDSSReportInstance::EvaluateView
IDSSReportInstance::Execute
IDSSReportInstance::ExecuteActions
IDSSReportInstance::ExecuteAsView
IDSSReportInstance::ExecuteFlags
IDSSReportInstance::ExecutionProgress
IDSSReportInstance::ExecutionStatus
IDSSReportInstance::Filter
IDSSReportInstance::HistoricalCost
IDSSReportInstance::JobID
IDSSReportInstance::LoadDrillFromXML
IDSSReportInstance::LoadFromXMLDOM
IDSSReportInstance::LookUpProperty
IDSSReportInstance::NewDrillInstance
IDSSReportInstance::OriginalDefinition
IDSSReportInstance::OriginalFilter
IDSSReportInstance::OriginalTemplate
IDSSReportInstance::PauseExecution
IDSSReportInstance::Project
IDSSReportInstance::RefreshState
IDSSReportInstance::ReportSource
IDSSReportInstance::Resolution
IDSSReportInstance::ResultSetSize
IDSSReportInstance::ResultUpdateTime
IDSSReportInstance::ResumeExecution
IDSSReportInstance::SaveAs
IDSSReportInstance::SaveGraphToFile
IDSSReportInstance::SaveToXMLDOM
IDSSReportInstance::Schema
IDSSReportInstance::SearchForPaths
IDSSReportInstance::SQLs
IDSSReportInstance::State
IDSSReportInstance::Template
IDSSReportInstance::User
IDSSReportInstance::UserSuppliedCost
IDSSReportInstance::UseSubsetReport
IDSSReportInstance::Views
IDSSReportInstance::XTabDataAccess
IDSSReportInstance2
IDSSReportInstance2::LookUpPropertyDefault
IDSSReportInstance2::LookUpPropertyLocation
IDSSReportInstance2::ReportDetails
IDSSReportInstance2::SaveGraphToMemory
IDSSReportInstance2::SaveReportResult
IDSSReportInstance3
IDSSReportInstance3::CacheID
IDSSReportInstance3::DatamartTableName
IDSSReportInstance3::GetPropertiesXML
IDSSReportInstance3::MessageID
IDSSReportInstance3::PutPropertiesXML
IDSSReportInstance3::ReportExecutionSettings
IDSSReportInstance4
IDSSReportInstance4::CachedFlag
IDSSReportInstance4::DrillByManipulation
IDSSReportInstance4::ExecutionInfo
IDSSReportInstance4::QueryStatistics
IDSSReportInstance4::ReportType
IDSSReportInstance4::WorkingSet
IDSSReportInstance5
IDSSReportInstance5::CalculateColumnWidth
IDSSReportInstance6
IDSSReportInstance6::InitialReportID
IDSSReportInstance7
IDSSReportInstance7::LookUpProperty2
IDSSReportInstance8
IDSSReportInstance8::DataFilter
IDSSReportInstance8::DataTemplate
IDSSReportInstance8::ExportContext
IDSSReportInstance8::ExportReport
IDSSReportInstance8::get_ViewTemplateByIndex
IDSSReportInstance8::IsWorkingset
IDSSReportInstance8::ParseMacros
IDSSReportInstance8::PromptInfo
IDSSReportInstance8::ReportProperties
IDSSReportInstance8::ViewFilter
IDSSReportInstance8::ViewTemplate
IDSSReportInstance9
IDSSReportInstance9::ParseMacros2
IDSSReportManipulation
IDSSReportManipulation::Accept
IDSSReportManipulation::AddDerivedMetric
IDSSReportManipulation::AddMetric
IDSSReportManipulation::
AddPercentToTotalMetric
IDSSReportManipulation::AddRankMetric
IDSSReportManipulation::
AddTransformationMetric
IDSSReportManipulation::AddUnit
IDSSReportManipulation::ApplyDeltaXML
IDSSReportManipulation::ApplyFix
IDSSReportManipulation::ApplyGraphDelta
IDSSReportManipulation::ApplyObjectLock
IDSSReportManipulation::ApplyStyle
IDSSReportManipulation::AxisSubtotals
IDSSReportManipulation::Cancel
IDSSReportManipulation::ChangeDisplayName
IDSSReportManipulation::ClearTemplate
IDSSReportManipulation::ClearTemplateUnit
IDSSReportManipulation::ConvertMetricInto
DerivedMetric
IDSSReportManipulation::FormatAxis
IDSSReportManipulation::FormatUnit
IDSSReportManipulation::HideBanding
IDSSReportManipulation::HideSubtotals
IDSSReportManipulation::HideThresholds
IDSSReportManipulation::MadeUp
IDSSReportManipulation::MakeNormal
IDSSReportManipulation::ManipulationType
IDSSReportManipulation::PageBy
IDSSReportManipulation::PivotUnit
IDSSReportManipulation::
RemoveInaccessibleMetrics
IDSSReportManipulation::RemoveSorting
IDSSReportManipulation::RemoveSubtotals
IDSSReportManipulation::RemoveUnit
IDSSReportManipulation::SetFilter
IDSSReportManipulation::SetTemplate
IDSSReportManipulation::Sort
IDSSReportManipulation::UnitSubtotals
IDSSReportManipulation::UpdateDerived
MetricExpression
IDSSReportManipulation::ValidationIssues
IDSSReportManipulation::ValidationLevel
IDSSReportManipulation2
IDSSReportManipulation2::PageBy2
IDSSReportManipulation2::SetWidthClear
IDSSReportManipulation2::SetWidthHeader
IDSSReportManipulation2::SetWidthMetric
IDSSReportManipulation2::SetWidthScenario
IDSSReportManipulation3
IDSSReportManipulation3::AddForm
IDSSReportManipulation3::RemoveForm
IDSSReportManipulation3::ResetForms
IDSSReportManipulation4
IDSSReportManipulation4::AddLimit
IDSSReportManipulation4::AddUnitToBase
IDSSReportManipulation4::FormatTemplate
IDSSReportManipulation4::RemoveLimit
IDSSReportManipulation5

IDSSReportManipulation5::ForceToNormal
IDSSReportManipulation5::
ForceToNormalForDatamart
IDSSReportManipulation5::SetDatamart
IDSSReportNetClient
IDSSReportNetClient::Init
IDSSReportNetClient::ProjectID
IDSSReportNetClient::RemoteTimeOut
IDSSReportNetClient::ServerNet
IDSSReportProperties
IDSSReportProperties::Add
IDSSReportProperties::Attributes
IDSSReportProperties::Clear
IDSSReportProperties::Count
IDSSReportProperties::Item
IDSSReportProperties::Name
IDSSReportProperties::Remove
IDSSReportServices
IDSSReportServices::GridFormatService
IDSSReportServices::ReportManipulationService
IDSSReportServices::StructureValidationService
IDSSReportServices::SubsetValidationService
IDSSReportServices::TextFormatService
IDSSReportServices::XMLCompressionService
IDSSReportServices2
IDSSReportServices2::GridFormatService2
IDSSReportServices2::RWGridXMLFormatter
IDSSReportServices2::StructureValidation
Service2
IDSSReportSource
IDSSReportSource::AnswerPrompt
IDSSReportSource::Execute
IDSSReportSource::ExecuteDefinition
IDSSReportSource::ExecuteFT
IDSSReportSource::ExecuteSQL
IDSSReportSource::GenerateAlerts
IDSSReportSource::GenerateSQL
IDSSReportSource::GetCacheAdmin
IDSSReportSource::MadeUP
IDSSReportSource::NewDrillInstance
IDSSReportSource::NewInstance
IDSSReportSource::NewInstanceFT
IDSSReportSource::RemoveCache
IDSSReportSource::Resolve
IDSSReportSource::XTab
IDSSReportSource2
IDSSReportSource2::CloneInstance
IDSSReportSource3
IDSSReportSource3::SetGraphUTF8API
IDSSReportSource4
IDSSReportSource4::PreExecute
IDSSReportSourceEvents
IDSSReportSourceEvents::ExecutionCancelled
IDSSReportSourceEvents::ExecutionComplete
IDSSReportSourceEvents::ExecutionFailed
IDSSReportSourceEvents::ProgressNotification
IDSSReportSourceEvents::ResolvePrompts
IDSSReportSourceServer
IDSSReportSourceServer::Init
IDSSReportSourceServer::PackageInstance
IDSSReportSourceServer::Process
IDSSReportSourceServer::ReportSource
IDSSReportSourceServer::Source
IDSSReportSourceServer2
IDSSReportSourceServer2::PackageInstance
IDSSReportValidator
IDSSReportValidator::ValidateSubsetMetric
IDSSReportValidator::ValidateSubsetReport
IDSSReportView
IDSSReportView::AttributeValues
IDSSReportView::GetResult
IDSSRequest
IDSSRequest::Executor
IDSSRequest::Info
IDSSRequest::Script
IDSSResolution
IDSSResolution::_NewEnum
IDSSResolution::Add
IDSSResolution::Apply
IDSSResolution::CleanUp
IDSSResolution::Clear
IDSSResolution::Combine
IDSSResolution::Count
IDSSResolution::Info
IDSSResolution::IsEquivalentAnswer
IDSSResolution::IsLessThan
IDSSResolution::Item
IDSSResolution::LoadAnswersFromXML
IDSSResolution::NextPrompt
IDSSResolution::Populate
IDSSResolution::Remove
IDSSResolution::Resolve
IDSSResolution2
IDSSResolution2::Answers
IDSSResolution3
IDSSResolution3::ResolveEx
IDSSResolution4
IDSSResolution4::CombineEx
IDSSResolution4::HasAutoClosePromptsOnly
IDSSResolutionServer
IDSSResolutionServer::Apply
IDSSResolutionServer::Execute
IDSSResolutionServer::Init
IDSSResolutionServer::Resolve
IDSSResolutionServer2
IDSSResolutionServer2::ResolveEx
IDSSRole
IDSSRole::ERType
IDSSRole::Info
IDSSRole::Populate
IDSSRole::Prefix
IDSSRole::RoleType
IDSSRoleMembership
IDSSRoleMembership::IsModified
IDSSRoleMembership::Membership
IDSSRoleMembership::Refresh
IDSSRoleMembership::SaveChanges
IDSSRoleMembership::SecurityRole
IDSSRoleMemberships
IDSSRoleMemberships::_NewEnum
IDSSRoleMemberships::Add
IDSSRoleMemberships::Clear
IDSSRoleMemberships::Count
IDSSRoleMemberships::FindRoles
IDSSRoleMemberships::IsModified
IDSSRoleMemberships::Item
IDSSRoleMemberships::Refresh
IDSSRoleMemberships::Remove
IDSSRoleMemberships::SaveChanges
IDSSRoles
IDSSRoles::_NewEnum
IDSSRoles::Add
IDSSRoles::Clear
IDSSRoles::Count
IDSSRoles::Item
IDSSRoles::Remove
IDSSRWDIssue
IDSSRWDIssue::CauseFormID
IDSSRWDIssue::CauseObjectID
IDSSRWDIssue::CauseObjectType
IDSSRWDIssue::FieldKey
IDSSRWDIssue::GetLocationNodeKey
IDSSRWDIssue::IsRecoverable
IDSSRWDIssue::IssueType
IDSSRWDIssues
IDSSRWDIssues::_NewEnum
IDSSRWDIssues::Count
IDSSRWDIssues::IsRecoverable
IDSSRWDIssues::Item
IDSSRWDocumentNode
IDSSRWDocumentNode::Document
IDSSRWDValidator
IDSSRWDValidator::ValidateRWDocument
IDSSRWDValidator2
IDSSRWDValidator2::ValidateRWDocument
Instance
IDSSRWDValidator3
IDSSRWDValidator3::ValidateRW
DocumentInstance
IDSSRWField
IDSSRWField::AttributeForms
IDSSRWField::AutoTextType
IDSSRWField::DataSourceObject
IDSSRWField::FieldObject
IDSSRWField::Fields
IDSSRWField::Format
IDSSRWField::Index
IDSSRWField::Key
IDSSRWField::Name
IDSSRWField::Text
IDSSRWField::Type
IDSSRWField::Value
IDSSRWField2
IDSSRWField2::Link
IDSSRWField3
IDSSRWField3::Expression
IDSSRWField3::Thresholds
IDSSRWFieldGroupNode
IDSSRWFieldGroupNode::DataLevel
IDSSRWFieldGroupNode::Fields
IDSSRWFieldGroupNode::JoinOption
IDSSRWFieldGroupNode::Separator
IDSSRWFieldGroupNode::SlicingOffObjects
IDSSRWFieldGroupNode::Sorts
IDSSRWFieldGroupNode2
IDSSRWFieldGroupNode2::FieldGroupLimit
IDSSRWFields
IDSSRWFields::_NewEnum
IDSSRWFields::Add
IDSSRWFields::AddCopy
IDSSRWFields::AddMove
IDSSRWFields::Clear
IDSSRWFields::Count
IDSSRWFields::Item
IDSSRWFields::Remove
IDSSRWGridXMLFormatter
IDSSRWGridXMLFormatter::SaveCSS
CollectionToXMLDOM
IDSSRWGridXMLFormatter::SaveRIToXMLDOM
IDSSRWIterator
IDSSRWIterator::Count
IDSSRWIterator::Item
IDSSRWIterator::Root
IDSSRWIterator::RWNode
IDSSRWIterator::Type
IDSSRWIterator2
IDSSRWIterator2::SpecialFGIterator
IDSSRWIterator2::Threshold
IDSSRWNode
IDSSRWNode::_NewEnum
IDSSRWNode::ChildIndex
IDSSRWNode::ChildNode
IDSSRWNode::Count
IDSSRWNode::FirstSibling
IDSSRWNode::Format
IDSSRWNode::Item
IDSSRWNode::Key
IDSSRWNode::LastSibling
IDSSRWNode::Next
IDSSRWNode::Options
IDSSRWNode::Parent
IDSSRWNode::Previous

IDSSRWNode::Root
IDSSRWNode::Special
IDSSRWNode::Tag
IDSSRWNode::Type
IDSSRWNode2
IDSSRWNode2::Thresholds
IDSSRWSectionIterator
IDSSRWSectionIterator::CountElements
IDSSRWSectionIterator::FirstElement
IDSSRWSectionIterator::GroupByElements
IDSSRWSectionIterator::NextElement
IDSSRWSectionIterator2
IDSSRWSectionIterator2::CountUnits
IDSSRWSectionIterator2::Text
IDSSRWSectionNode
IDSSRWSectionNode::Add
IDSSRWSectionNode::AddCopy
IDSSRWSectionNode::AddMove
IDSSRWSectionNode::Clear
IDSSRWSectionNode::GroupByUnits
IDSSRWSectionNode::Remove
IDSSRWSectionNode::SectionType
IDSSRWSectionNode2
IDSSRWSectionNode2::PageByStyle
IDSSRWSectionNode2::SectionLimit
IDSSRWTemplateNode
IDSSRWTemplateNode::DataSourceObject
IDSSRWTemplateNode::DisplayMode
IDSSRWTemplateNode::Template
IDSSRWTemplateNode2
IDSSRWTemplateNode2::GridColumnLimit
IDSSRWTemplateNode2::GridRowLimit
IDSSScheduleAdministrator
IDSSScheduleAdministrator::FireEvent
IDSSScheduleAdministrator::
LoadScheduleObject
IDSSScheduleAdministrator::
LoadScheduleObjects
IDSSScheduleAdministrator::
ScheduleInstances
IDSSScheduleAdministrator::
UnLoadScheduleInstances
IDSSScheduleAdministrator2
IDSSScheduleAdministrator2::CopySchedule
IDSSScheduleAdministrator2::
GetPersonalization
IDSSScheduleAdministrator2::
GetServerLocalTime
IDSSScheduleAdministrator2::
PopulateStatement
IDSSScheduleAdministrator2::Schedule
IDSSScheduleAdministrator2::
ScheduleInstancesAll
IDSSScheduleAdministrator2::ScheduleInstances
Configuration
IDSSScheduleAdministrator2::Schedule
InstancesDocument
IDSSScheduleAdministrator2::
ScheduleInstancesGeneral
IDSSScheduleAdministrator2::
ScheduleInstancesReport
IDSSScheduleAdministrator2::Unschedule
IDSSScheduleAdministrator2::UnscheduleAll
IDSSScheduleAdministrator2::
UnscheduleConfiguration
IDSSScheduleAdministrator2::
UnscheduleDocument
IDSSScheduleAdministrator2::
UnscheduleGeneral
IDSSScheduleAdministrator2::
UnscheduleReport
IDSSScheduleEvent
IDSSScheduleEvent::Info
IDSSScheduleInstance
IDSSScheduleInstance::EventID
IDSSScheduleInstance::EventName
IDSSScheduleInstance::ID
IDSSScheduleInstance::Name
IDSSScheduleInstance::NextExecTime
IDSSScheduleInstance::ProjectID
IDSSScheduleInstance::ProjectName
IDSSScheduleInstance::RequestID
IDSSScheduleInstance::RequestName
IDSSScheduleInstance::TriggerID
IDSSScheduleInstance::TriggerName
IDSSScheduleInstance::Type
IDSSScheduleInstance::Unload
IDSSScheduleInstance::UserID
IDSSScheduleInstance::UserName
IDSSScheduleInstances
IDSSScheduleInstances::_NewEnum
IDSSScheduleInstances::Count
IDSSScheduleInstances::Item
IDSSScheduleInstances::ItemByIndex
IDSSScheduleInstances::ItemByName
IDSSScheduleObject
IDSSScheduleObject::Info
IDSSScheduleObject::Request
IDSSScheduleObject::ScheduleTrigger
IDSSScheduleTrigger
IDSSScheduleTrigger::EndTime
IDSSScheduleTrigger::Event
IDSSScheduleTrigger::Info
IDSSScheduleTrigger::MaxOccurrences
IDSSScheduleTrigger::OccurrencesLeft
IDSSScheduleTrigger::ScheduleTriggerType
IDSSScheduleTrigger::StartTime
IDSSScheduleTrigger::TimePointSeries
IDSSSchema
IDSSSchema::Attributes
IDSSSchema::Columns
IDSSSchema::Dimensions
IDSSSchema::FactGroups
IDSSSchema::Facts
IDSSSchema::Forms
IDSSSchema::Functions
IDSSSchema::Info
IDSSSchema::Populate
IDSSSchema::Roles
IDSSSchema::SystemDimension
IDSSSchema::Tables
IDSSSchema2
IDSSSchema2::AttributeInfos
IDSSSchema2::Catalog
IDSSSchema2::DBRole
IDSSSchema2::DimensionInfos
IDSSSchema2::DrillMap
IDSSSchema2::DrillMapForObject
IDSSSchema2::LanguageCode
IDSSSchema2::NextDrillMapForObject
IDSSSchema2::UnitDrillMap
IDSSSchema2::XDADefinition
IDSSSchema2::XDAMetrics
IDSSSchema2::XDAType
IDSSSchemaServices
IDSSSchemaServices::CopyObject
IDSSSchemaServices::DeleteObject
IDSSSchemaServices::NewObject
IDSSScript
IDSSScript::Statement
IDSSScriptProcessor
IDSSScriptProcessor::CreateObjects
IDSSScriptProcessor::CreateObjectsInCollection
IDSSScriptProcessor::GetObjectInfo
IDSSScriptProcessor::Load
IDSSScriptProcessor::ObjectCount
IDSSScriptProcessor::PopulateSQLPatterns
IDSSScriptProcessor::SymbolID
IDSSScriptServer
IDSSScriptServer::CancelRequest
IDSSScriptServer::GetRequestResult
IDSSScriptServer::ReportView
IDSSScriptServer::SendRequest
IDSSScriptServer2
IDSSScriptServer2::GetRequestResultIn
ReportInstances
IDSSScriptServer3
IDSSScriptServer3::GetRequestResultIn
Instances
IDSSScriptServer3::SendBatchExecutionRequest
IDSSScriptServer4
IDSSScriptServer4::SendBatchExecution
Request2
IDSSScriptServer5
IDSSScriptServer5::SessionID
IDSSSearch
IDSSSearch::AbbreviationPattern
IDSSSearch::AccessRights
IDSSSearch::BeginTime
IDSSSearch::Clear
IDSSSearch::DescriptionPattern
IDSSSearch::Domain
IDSSSearch::EndTime
IDSSSearch::Execute
IDSSSearch::Flags
IDSSSearch::GetSID
IDSSSearch::Info
IDSSSearch::IntervalUnit
IDSSSearch::MaxObjects
IDSSSearch::NamePattern
IDSSSearch::Populate
IDSSSearch::PropertyRestrictions
IDSSSearch::PutSID
IDSSSearch::SearchOwner
IDSSSearch::SearchRoot
IDSSSearch::StateClear
IDSSSearch::StateSet
IDSSSearch::TimeInterval
IDSSSearch::Types
IDSSSearch::UsedBy
IDSSSearch::Uses
IDSSSearch2
IDSSSearch2::Scope
IDSSSecurityRole
IDSSSecurityRole::Info
IDSSSecurityRole::Privileges
IDSSServerAdmin
IDSSServerAdmin::ApplyRunTimeSettings
IDSSServerAdmin::CacheAdministrator
IDSSServerAdmin::ChangeUserPassword
IDSSServerAdmin::CloseUserSession
IDSSServerAdmin::ClusterManager
IDSSServerAdmin::ConfigurationID
IDSSServerAdmin::Create
IDSSServerAdmin::DatabaseAdministrator
IDSSServerAdmin::DataSource
IDSSServerAdmin::DBMonitors
IDSSServerAdmin::DiagLogFlag
IDSSServerAdmin::DisconnectUser
IDSSServerAdmin::GetUserInBox
IDSSServerAdmin::Idle
IDSSServerAdmin::JobManager
IDSSServerAdmin::LoginProject
IDSSServerAdmin::LogoutProject
IDSSServerAdmin::LogoutUserFromProject
IDSSServerAdmin::Name
IDSSServerAdmin::Ping
IDSSServerAdmin::ProjectInstances
IDSSServerAdmin::RefreshSQLEngineSchema
IDSSServerAdmin::RefreshUserObjects
IDSSServerAdmin::RunRequest
IDSSServerAdmin::RunRequestByID

IDSSServerAdmin::SaveInBoxToFile
IDSSServerAdmin::ScheduleAdministrator
IDSSServerAdmin::ScheduleInstances
IDSSServerAdmin::ServerID
IDSSServerAdmin::ServerMemAdministrator
IDSSServerAdmin::ServerNet
IDSSServerAdmin::ServerSummary
IDSSServerAdmin::ServiceName
IDSSServerAdmin::Session
IDSSServerAdmin::Start
IDSSServerAdmin::Statistics
IDSSServerAdmin::Status
IDSSServerAdmin::Stop
IDSSServerAdmin::UpdateDiagnostics
IDSSServerAdmin::UserSessions
IDSSServerAdmin2
IDSSServerAdmin2::UserProjectInstances
IDSSServerAdmin3
IDSSServerAdmin3::AvailablePrivilegeBitMask
IDSSServerAdmin3::IsAvailablePrivilege
IDSSServerAdmin4
IDSSServerAdmin4::LicenseComplianceManager
IDSSServerAdmin5
IDSSServerAdmin5::IsNamedUserAvailable Privilege
IDSSServerAdmin5::NamedUserAvailable PrivilegeBitMask
IDSSServerAdmin6
IDSSServerAdmin6::ListOfODBCDSNs
IDSSServerAdmin6::PerfMonitorEnabled
IDSSServerAdmin7
IDSSServerAdmin7::ClearXSLCache
IDSSServerDef
IDSSServerDef::Channels
IDSSServerDef::Configuration
IDSSServerDef::InboxFilePath
IDSSServerDef::InboxWriteToDiskFreq
IDSSServerDef::Info
IDSSServerDef::JobPriorityMap
IDSSServerDef::MaxInboxMsgCountPerUser Session
IDSSServerDef::ObjectBucketPerLock
IDSSServerDef::ObjectCacheFilePath
IDSSServerDef::ObjectCacheTableBucketCount
IDSSServerDef::ObjectMaxCacheCount
IDSSServerDef::ObjectMaxCacheSize
IDSSServerDef::ObjectReplacementPolicy
IDSSServerDef::Populate
IDSSServerDef::ProcessingUnits
IDSSServerDef::ProjectReferences
IDSSServerDef::Setting
IDSSServerDef::SettingCount
IDSSServerDef::Tasks
IDSSServerDef2
IDSSServerDef2::IsInternalUsage
IDSSServerDefManipulator
IDSSServerDefManipulator::ApplyDeltaXML
IDSSServerDefManipulator::RegisterProject
IDSSServerDefManipulator::ServerDef
IDSSServerDefManipulator:: SetProjectLoadOption
IDSSServerDefManipulator::SetProjectSetting
IDSSServerDefManipulator::SetServerSetting
IDSSServerDefManipulator::UnregisterProject
IDSSServerDefManipulator::UserRuntime
IDSSServerMemAdmin
IDSSServerMemAdmin::AllHeapsByteCount
IDSSServerMemAdmin::Checkpoint
IDSSServerMemAdmin:: DiffSnapshotNowFromLast
IDSSServerMemAdmin::GetAllocations
IDSSServerMemAdmin::MemMisc
IDSSServerMemAdmin::NTCounter
IDSSServerMemAdmin::Refresh
IDSSServerMemAdmin::RefreshNTCounter
IDSSServerMemAdmin::RefreshServerMemMisc
IDSSServerMemAdmin::ReportAllocations
IDSSServerMemAdmin:: SetAllocationCheckPoint
IDSSServerMemAdmin::StartAllocationLog
IDSSServerMemAdmin::StopAllocationLog
IDSSServerMemAdmin::TakeHeapSnapshot
IDSSServerMemAdmin2
IDSSServerMemAdmin2::AddNewCheckPoint
IDSSServerMemAdmin2::DumpAllUsageLeak
IDSSServerMemAdmin2::DumpUsageLeak
IDSSServerMemAdmin2::SetLogFileName
IDSSServerMemAdmin2::StartCallStackTracing
IDSSServerMemAdmin2::StopCallStackTracing
IDSSServerMemAdmin3
IDSSServerMemAdmin3:: GetObjectInstanceCount
IDSSServerMemAdmin4
IDSSServerMemAdmin4::GetServrInfoCount
IDSSServerMessage
IDSSServerMessage::ClientType
IDSSServerMessage::FinishTime
IDSSServerMessage::FolderName
IDSSServerMessage::MsgCreationTime
IDSSServerMessage::MsgID
IDSSServerMessage::MsgModificationTime
IDSSServerMessage::MsgStatus
IDSSServerMessage::MsgText
IDSSServerMessage::MsgTitle
IDSSServerMessage::MsgType
IDSSServerMessage::ObjCreatorName
IDSSServerMessage::ObjDescription
IDSSServerMessage::ObjectID
IDSSServerMessage::ParentMsgID
IDSSServerMessage::ProjectID
IDSSServerMessage::ProjectName
IDSSServerMessage::RqstStatus
IDSSServerMessage::RqstType
IDSSServerMessage::StartTime
IDSSServerMessages
IDSSServerMessages::_NewEnum
IDSSServerMessages::ChangeItemStatus
IDSSServerMessages::Count
IDSSServerMessages::DeleteItem
IDSSServerMessages::Item
IDSSServerMessages::ItemByIndex
IDSSServerNet
IDSSServerNet::AdviseMsgCB
IDSSServerNet::CheckVersion
IDSSServerNet::Connect
IDSSServerNet::Disconnect
IDSSServerNet::FreeThreadingAdviseMsgCB
IDSSServerNet::GetStatus
IDSSServerNet::Init
IDSSServerNet::InitWithAnonymous Authentication
IDSSServerNet::InitWithIntegrateAuthentication
IDSSServerNet::InitWithProjectDatabase Authentication
IDSSServerNet::InitWithStandardAuthentication
IDSSServerNet::NewMsg
IDSSServerNet::NewRequestMsg
IDSSServerNet::Send
IDSSServerNet::SendAndReceive
IDSSServerNet::Timeout
IDSSServerNet::UnAdviseMsgCB
IDSSServerNet2
IDSSServerNet2::ProjectName
IDSSServerNet2::ServerContext
IDSSServerNetSink
IDSSServerNetSink::MsgCB
IDSSServerSummary
IDSSServerSummary:: RefreshDSSServerSummary
IDSSServerSummary::ServerSummary
IDSSServerSummary2
IDSSServerSummary2::ServerSummaryValue
IDSSSession
IDSSSession::ChangeUserPassword
IDSSSession::CheckUserPrivilege
IDSSSession::ClientServices
IDSSSession::Close
IDSSSession::Component
IDSSSession::DataSource
IDSSSession::ElementSource
IDSSSession::InBox
IDSSSession::ObjectSource
IDSSSession::ReportSource
IDSSSession::ServerAdmin
IDSSSession::ServerNet
IDSSSession::User
IDSSSession::UserName
IDSSSession::UserRuntime
IDSSShortcut
IDSSShortcut::Exports
IDSSShortcut::IsMe
IDSSShortcut::Node
IDSSShortcut::Populate
IDSSShortcut::Prompt
IDSSShortcut::Target
IDSSShortcut::TargetID
IDSSShortcut::TargetName
IDSSShortcut::TargetPath
IDSSShortcut::TargetType
IDSSShortcut2
IDSSShortcut2::DataSourceObject
IDSSShortcut2::SubstituteFunctionType
IDSSShortcutObject
IDSSShortcutObject::Info
IDSSShortcutObject::Populate
IDSSShortcutObject::Target
IDSSShortcutObject::TargetID
IDSSShortcutObject::TargetName
IDSSShortcutObject::TargetType
IDSSSlicedGridWindow
IDSSSlicedGridWindow::GetWindow
IDSSSlicedGridWindow::WindowSize
IDSSSort
IDSSSort::Ascending
IDSSSort::Attribute
IDSSSort::Consolidation
IDSSSort::CustomGroup
IDSSSort::Dimension
IDSSSort::Elements
IDSSSort::Form
IDSSSort::Index
IDSSSort::Locale
IDSSSort::Metric
IDSSSort::SortType
IDSSSort::Subtotal
IDSSSort::SubtotalsPosition
IDSSSort2
IDSSSort2::IsParentFirst
IDSSSort3
IDSSSort3::CustomGroupSortType
IDSSSorts
IDSSSorts::_NewEnum
IDSSSorts::Add
IDSSSorts::Clear
IDSSSorts::Count
IDSSSorts::Item
IDSSSorts::Order
IDSSSorts::Remove
IDSSSource
IDSSSource::Administrator
IDSSSource::Configuration
IDSSSource::CopyObject

IDSSSource::DeleteObject
IDSSSource::ExecuteCommand
IDSSSource::ExecuteSearch
IDSSSource::FindObject
IDSSSource::LoadSchema
IDSSSource::MoveObject
IDSSSource::NameScope
IDSSSource::NewObject
IDSSSource::Project
IDSSSource::RefreshObject
IDSSSource::ReplaceDependents
IDSSSource::Root
IDSSSource::SaveObject
IDSSSource::Schema
IDSSSource::Separator
IDSSSource2
IDSSSource2::GetPropertiesXML
IDSSSource2::LockObject
IDSSSource2::PutPropertiesXML
IDSSSource2::QueryObjectLock
IDSSSource2::UnlockObject
IDSSSource3
IDSSSource3::DeleteObjects
IDSSSource3::SaveObjects
IDSSSource4
IDSSSource4::DeleteLinkItemsXML
IDSSSource4::FindLinkItemsXML
IDSSSource4::SaveLinkItemsXML
IDSSSource5
IDSSSource5::CompareObjects
IDSSSource5::IsSystemObject
IDSSSource5::SynchronizeVersion
IDSSSourceEvents
IDSSSourceEvents::DeleteObjectComplete
IDSSSourceEvents::ExecuteSearchComplete
IDSSSourceEvents::FindObjectComplete
IDSSSourceEvents::LoadObjectComplete
IDSSSourceEvents::LoadSchemaComplete
IDSSSourceEvents::ProgressReport
IDSSSourceNetClient
IDSSSourceNetClient::ContentSource
IDSSSourceNetClient::Init
IDSSSourceNetClient::ServerNet
IDSSSourceServer
IDSSSourceServer::Source
IDSSSQL
IDSSSQL::DisplayCleanup
IDSSSQL::DisplaySQL
IDSSSQL::Statement
IDSSSQLEngineServer
IDSSSQLEngineServer::Execute
IDSSSQLEngineServer::Init
IDSSSQLEngineServer::ResetSchema
IDSSSQLs
IDSSSQLs::_NewEnum
IDSSSQLs::Add
IDSSSQLs::Count
IDSSSQLs::Item
IDSSSQLsServer
IDSSSQLsServer::NewSQLs
IDSSStatement
IDSSStatement::CommandID
IDSSStatement::Name
IDSSStatement::Parameter
IDSSStatement::ParameterCount
IDSSStatement::ParametersMode
IDSSStatement::StatementType
IDSSStatement::Substatements
IDSSStatement2
IDSSStatement2::User
IDSSStatements
IDSSStatements::_NewEnum
IDSSStatements::Add
IDSSStatements::Clear
IDSSStatements::Count
IDSSStatements::Item
IDSSStatements::Remove
IDSSStatistics
IDSSStatistics::GetCollectionSetting
IDSSStatistics::Purge
IDSSStatistics::SetCollectionSetting
IDSSStatistics2
IDSSStatistics2::GetStatsVersion
IDSSStatistics2::IsStatsRunning
IDSSStatistics2::Purge2
IDSSStatistics3
IDSSStatistics3::Purge3
IDSSStructureValidation
IDSSStructureValidation::BaseObject
IDSSStructureValidation::ValidationIssues
IDSSStructureValidation::ValidationLevel
IDSSStyle
IDSSStyle::Apply
IDSSStyle::Info
IDSSStyle::Populate
IDSSStyle::PopulateFromTemplate
IDSSSubsetValidation
IDSSSubsetValidation::BaseReport
IDSSSubsetValidation::SubsetReport
IDSSSubsetValidation::ValidationIssues
IDSSSubsetValidation::ValidationLevel
IDSSSyntaxServer
IDSSSyntaxServer::CatalogTableName
IDSSSyntaxServer::DropTableStmt
IDSSTable
IDSSTable::AttributeInfos
IDSSTable::DbTable
IDSSTable::Extensions
IDSSTable::FactGroups
IDSSTable::FactInfos
IDSSTable::HomogeneousDefn
IDSSTable::Info
IDSSTable::IsPartOfPartition
IDSSTable::IsTrueKey
IDSSTable::PartitionFilter
IDSSTable::PartitionLevel
IDSSTable::PartitionTables
IDSSTable::Populate
IDSSTable::PrimaryDBRole
IDSSTable::PrimaryTable
IDSSTable::SecondaryDBRoles
IDSSTable::Size
IDSSTable::TableKey
IDSSTable::TableType
IDSSTable2
IDSSTable2::DimensionInfos
IDSSTable2::Dimensions
IDSSTable2::MetricInfos
IDSSTable2::Scope
IDSSTable2::SharedAttributeInfos
IDSSTable2::TableDimension
IDSSTable2::XDADefinition
IDSSTable2::XDASchema
IDSSTable2::XDAType
IDSSTable3
IDSSTable3::AllAttributeInfos
IDSSTable3::ImpliedAttributeInfos
IDSSTable3::ImpliedDimensionInfos
IDSSTable3::MDXSAPVariableInfos
IDSSTables
IDSSTables::_NewEnum
IDSSTables::Add
IDSSTables::Clear
IDSSTables::Count
IDSSTables::Item
IDSSTables::Remove
IDSSTableSource
IDSSTableSource::Info
IDSSTableSource::Populate
IDSSTableSource::Prefix
IDSSTableSource::Suffix
IDSSTableSources
IDSSTableSources::_NewEnum
IDSSTableSources::Add
IDSSTableSources::Clear
IDSSTableSources::Count
IDSSTableSources::Item
IDSSTableSources::Remove
IDSSTask
IDSSTask::Description
IDSSTask::Name
IDSSTasks
IDSSTasks::_NewEnum
IDSSTasks::Add
IDSSTasks::Clear
IDSSTasks::Count
IDSSTasks::Item
IDSSTasks::Remove
IDSSTemplate
IDSSTemplate::ApplyDeltaXML
IDSSTemplate::Axes
IDSSTemplate::Columns
IDSSTemplate::CustomSubtotals
IDSSTemplate::Data
IDSSTemplate::DrillMap
IDSSTemplate::Filter
IDSSTemplate::FilterExports
IDSSTemplate::Info
IDSSTemplate::IsActive
IDSSTemplate::IsSubset
IDSSTemplate::MergeSubset
IDSSTemplate::Metrics
IDSSTemplate::PageOrdinal
IDSSTemplate::Pages
IDSSTemplate::Populate
IDSSTemplate::PropertyTargets
IDSSTemplate::Rows
IDSSTemplate::Subtotals
IDSSTemplate::SubtractBase
IDSSTemplate::Units
IDSSTemplate2
IDSSTemplate2::DataFormat
IDSSTemplate2::DisabledMetrics
IDSSTemplate2::HeaderFormat
IDSSTemplate2::HeaderValuesFormat
IDSSTemplate2::IsHideSubtotals
IDSSTemplate2::ReportFormat
IDSSTemplate2::SubtotalHeadersFormat
IDSSTemplate2::SubtotalValuesFormat
IDSSTemplate3
IDSSTemplate3::DisableRuntimeUnits
IDSSTemplate3::EnableRuntimeUnits
IDSSTemplate3::RemoveRuntimeInfo
IDSSTemplate3::SlicedGridWindow
IDSSTemplateAttribute
IDSSTemplateAttribute::Attribute
IDSSTemplateAttribute::AttributePrompt
IDSSTemplateAttribute::Dimension
IDSSTemplateAttribute::Forms
IDSSTemplateAttribute::PartialType
IDSSTemplateAttribute::TemplateUnit
IDSSTemplateAttribute::UseDefault
IDSSTemplateAttribute2
IDSSTemplateAttribute2::FormDataStatus
IDSSTemplateAttributes
IDSSTemplateAttributes::_NewEnum
IDSSTemplateAttributes::Count
IDSSTemplateAttributes::Item
IDSSTemplateConsolidation
IDSSTemplateConsolidation::Consolidation
IDSSTemplateConsolidation::
ConsolidationPrompt

IDSSTemplateConsolidation::Depth
IDSSTemplateConsolidation::EvaluationOrder
IDSSTemplateConsolidation::Flatten
IDSSTemplateConsolidation::Focus
IDSSTemplateConsolidation::Height
IDSSTemplateConsolidation::TemplateUnit
IDSSTemplateConsolidation2
IDSSTemplateConsolidation2::Aggregation
IDSSTemplateCustomGroup
IDSSTemplateCustomGroup::CustomGroup
IDSSTemplateCustomGroup::
CustomGroupPrompt
IDSSTemplateCustomGroup::Depth
IDSSTemplateCustomGroup::Flatten
IDSSTemplateCustomGroup::Forms
IDSSTemplateCustomGroup::Height
IDSSTemplateCustomGroup::TemplateUnit
IDSSTemplateCustomGroup::UseDefault
IDSSTemplateCustomGroup2
IDSSTemplateCustomGroup2::Aggregation
IDSSTemplateCustomGroup2::LocalFlatten
IDSSTemplateDimension
IDSSTemplateDimension::Dimension
IDSSTemplateDimension::DimensionPrompt
IDSSTemplateDimension::Forms
IDSSTemplateDimension::TemplateAttributes
IDSSTemplateDimension::TemplateUnit
IDSSTemplateDimension::UseDefault
IDSSTemplateElement
IDSSTemplateElement::Depth
IDSSTemplateElement::MergeHeader
IDSSTemplateElement2
IDSSTemplateElement2::ValueAndStatus
IDSSTemplateMetric
IDSSTemplateMetric::Alias
IDSSTemplateMetric::EvaluationOrder
IDSSTemplateMetric::Index
IDSSTemplateMetric::IsDecomposed
IDSSTemplateMetric::MakeCurrentElement
IDSSTemplateMetric::Metric
IDSSTemplateMetric::MetricExports
IDSSTemplateMetric::MetricLimit
IDSSTemplateMetric::MetricPrompt
IDSSTemplateMetric2
IDSSTemplateMetric2::GridFormat
IDSSTemplateMetric2::HeaderFormat
IDSSTemplateMetric2::IsDisabled
IDSSTemplateMetric2::MetricGridFormat
IDSSTemplateMetric2::MetricHeaderFormat
IDSSTemplateMetric2::ObjectLock
IDSSTemplateMetric2::Thresholds
IDSSTemplateMetric3
IDSSTemplateMetric3::ColumnWidth
IDSSTemplateMetric4
IDSSTemplateMetric4::DataStatus
IDSSTemplateMetrics
IDSSTemplateMetrics::_NewEnum
IDSSTemplateMetrics::Add
IDSSTemplateMetrics::Count
IDSSTemplateMetrics::EvaluationOrder
IDSSTemplateMetrics::Item
IDSSTemplateMetrics::MetricLimit
IDSSTemplateMetrics::MetricLimitType
IDSSTemplateMetrics::MetricsPrompt
IDSSTemplateMetrics::Remove
IDSSTemplateMetrics::TemplateUnit
IDSSTemplateMetrics2
IDSSTemplateMetrics2::ClearDisabled
IDSSTemplateMetrics2::DisabledCount
IDSSTemplateMetrics2::DisabledItem
IDSSTemplateMetrics2::RemoveDisabled
IDSSTemplates
IDSSTemplates::_NewEnum
IDSSTemplates::Add
IDSSTemplates::Clear
IDSSTemplates::Count
IDSSTemplates::EvaluateItem
IDSSTemplates::Item
IDSSTemplates::Remove
IDSSTemplateSubtotal
IDSSTemplateSubtotal::_NewEnum
IDSSTemplateSubtotal::Add
IDSSTemplateSubtotal::AxisSubtotals
IDSSTemplateSubtotal::Clear
IDSSTemplateSubtotal::Count
IDSSTemplateSubtotal::Definition
IDSSTemplateSubtotal::EvaluationOrder
IDSSTemplateSubtotal::Index
IDSSTemplateSubtotal::Item
IDSSTemplateSubtotal::Name
IDSSTemplateSubtotal::Remove
IDSSTemplateSubtotal::SubtotalType
IDSSTemplateSubtotals
IDSSTemplateSubtotals::_NewEnum
IDSSTemplateSubtotals::Add
IDSSTemplateSubtotals::Clear
IDSSTemplateSubtotals::Count
IDSSTemplateSubtotals::Item
IDSSTemplateSubtotals::Remove
IDSSTemplateUnit
IDSSTemplateUnit::Axis
IDSSTemplateUnit::AxisIndex
IDSSTemplateUnit::CurrentElement
IDSSTemplateUnit::CurrentElements
IDSSTemplateUnit::DrillMap
IDSSTemplateUnit::HeaderCount
IDSSTemplateUnit::Index
IDSSTemplateUnit::IsGrouped
IDSSTemplateUnit::IsNormalOrder
IDSSTemplateUnit::Name
IDSSTemplateUnit::Prompt
IDSSTemplateUnit::Special
IDSSTemplateUnit::Subtotals
IDSSTemplateUnit::UnitElements
IDSSTemplateUnit::UnitType
IDSSTemplateUnit2
IDSSTemplateUnit2::Alias
IDSSTemplateUnit2::CurrentElementID
IDSSTemplateUnit2::GridFormat
IDSSTemplateUnit2::HeaderFormat
IDSSTemplateUnit2::ObjectLock
IDSSTemplateUnit2::SubtotalGridFormat
IDSSTemplateUnit2::SubtotalHeaderFormat
IDSSTemplateUnit3
IDSSTemplateUnit3::ColumnWidths
IDSSTemplateUnit3::CurrentElementStatus
IDSSTemplateUnit5
IDSSTemplateUnit5::DataStatus
IDSSTemplateUnit5::UnitLimit
IDSSTemplateUnits
IDSSTemplateUnits::_NewEnum
IDSSTemplateUnits::Count
IDSSTemplateUnits::Item
IDSSTestResolution
IDSSTestResolution::TestPartialResolution
IDSSTextConvertor
IDSSTextConvertor::DoubleToText
IDSSTextConvertor::DoubleToTextH
IDSSTextConvertor::GetFormatHandle
IDSSTextConvertor::LongToText
IDSSTextConvertor::LongToTextH
IDSSTextConvertor::TextToDouble
IDSSTextConvertor::TextToLong
IDSSTextFormatter
IDSSTextFormatter::CSVFormattedResult
IDSSThreshold
IDSSThreshold::Condition
IDSSThreshold::Format
IDSSThreshold::Index
IDSSThreshold::IsEnabled
IDSSThreshold::Name
IDSSThreshold::Populate
IDSSThreshold::Prefix
IDSSThreshold::ReplaceMetric
IDSSThreshold::ReplaceText
IDSSThreshold::Semantics
IDSSThreshold::Suffix
IDSSThreshold2
IDSSThreshold2::_NewEnum
IDSSThreshold2::Add
IDSSThreshold2::Clear
IDSSThreshold2::Count
IDSSThreshold2::Item
IDSSThreshold2::LevelType
IDSSThreshold2::Remove
IDSSThreshold2::Scope
IDSSThreshold2::Subtotal
IDSSThresholdIterator
IDSSThresholdIterator::Column
IDSSThresholdIterator::Metric
IDSSThresholdIterator::Next
IDSSThresholdIterator::Reset
IDSSThresholdIterator::Row
IDSSThresholdIterator::Threshold
IDSSThresholds
IDSSThresholds::_NewEnum
IDSSThresholds::Add
IDSSThresholds::Clear
IDSSThresholds::Count
IDSSThresholds::Item
IDSSThresholds::Remove
IDSSThresholds::SetIsEnabled
IDSSTime
IDSSTime::DateTime
IDSSTime::Day
IDSSTime::DayOfWeek
IDSSTime::Difference
IDSSTime::DynamicDayOffset
IDSSTime::DynamicTime
IDSSTime::Format
IDSSTime::GetNthTimePointBefore
IDSSTime::GetShift
IDSSTime::Hour
IDSSTime::Init
IDSSTime::IsEqualTo
IDSSTime::IsGmt
IDSSTime::IsGreaterThan
IDSSTime::IsLessThan
IDSSTime::MilliSeconds
IDSSTime::Minutes
IDSSTime::Mode
IDSSTime::Month
IDSSTime::Name
IDSSTime::PopulateDefn
IDSSTime::Seconds
IDSSTime::Shift
IDSSTime::Text
IDSSTime::Week
IDSSTime::Year
IDSSTimeNode
IDSSTimeNode::Node
IDSSTimeNode::Prompt
IDSSTimeNode::Time
IDSSTimePeriod
IDSSTimePeriod::Days
IDSSTimePeriod::Hours
IDSSTimePeriod::MilliSeconds
IDSSTimePeriod::Minutes
IDSSTimePeriod::Months
IDSSTimePeriod::Seconds
IDSSTimePeriod::Weeks
IDSSTimePeriod::Years

IDSSTimePointSeries
IDSSTimePointSeries::Day
IDSSTimePointSeries::DayOfWeek
IDSSTimePointSeries::DayOfWeekRecurrence
IDSSTimePointSeries::DayRecurrence
IDSSTimePointSeries::GetAllTimePoints
IDSSTimePointSeries::GetNextNTimePoints
IDSSTimePointSeries::Hour
IDSSTimePointSeries::HourRecurrence
IDSSTimePointSeries::Minute
IDSSTimePointSeries::MinuteRecurrence
IDSSTimePointSeries::Month
IDSSTimePointSeries::MonthRecurrence
IDSSTimePointSeries::Second
IDSSTimePointSeries::SecondRecurrence
IDSSTimePointSeries::Week
IDSSTimePointSeries::WeekRecurrence
IDSSTimePointSeries::Window
IDSSTimePointSeries::Year
IDSSTimePointSeries::YearRecurrence
IDSSTimes
IDSSTimes::_NewEnum
IDSSTimes::Add
IDSSTimes::Clear
IDSSTimes::Count
IDSSTimes::Item
IDSSTimes::Remove
IDSSToken
IDSSToken::Context
IDSSToken::Extra
IDSSToken::Length
IDSSToken::Level
IDSSToken::Object
IDSSToken::ObjectType
IDSSToken::Populate
IDSSToken::StartPosition
IDSSToken::State
IDSSToken::Tag
IDSSToken::Type
IDSSToken::Value
IDSSTokenStream
IDSSTokenStream::_NewEnum
IDSSTokenStream::Add
IDSSTokenStream::Clear
IDSSTokenStream::Count
IDSSTokenStream::Insert
IDSSTokenStream::Item
IDSSTokenStream::Remove
IDSSTokenStream::Text
IDSSTypeMapping
IDSSTypeMapping::ConvertDSSToNative
IDSSTypeMapping::ConvertNativeToDSS
IDSSTypeMapping::SetDSSToNative
IDSSTypeMapping::SetNativeToDSS
IDSSTypes
IDSSTypes::_NewEnum
IDSSTypes::Add
IDSSTypes::Clear
IDSSTypes::Count
IDSSTypes::Item
IDSSTypes::Remove
IDSSUser
IDSSUser::Info
IDSSUser::IsGroup
IDSSUser::Membership
IDSSUser::Populate
IDSSUser::UserAccount
IDSSUserAccount
IDSSUserAccount::CopyMembership
IDSSUserAccount::Enabled
IDSSUserAccount::GetWindowsNTSID
IDSSUserAccount::InBox
IDSSUserAccount::Info
IDSSUserAccount::Membership
IDSSUserAccount::PasswordExpirationDate
IDSSUserAccount::PasswordExpirationDelta
IDSSUserAccount::PasswordExpires Automatically
IDSSUserAccount::PasswordModifiable
IDSSUserAccount::Privileges
IDSSUserAccount::ProjectAccessible
IDSSUserAccount::RDBMSLogin
IDSSUserAccount::RequiresNewPassword
IDSSUserAccount::SetPassword
IDSSUserAccount::SetWindowsNTSID
IDSSUserAccount::StandardAuthAllowed
IDSSUserAccount::User
IDSSUserAccount::UserSecurityFilters
IDSSUserAccount2
IDSSUserAccount2::LDAPDN
IDSSUserAccount2::PasswordExpiration Frequency
IDSSUserAccount2::SetRDBMSPassword
IDSSUserAccount3
IDSSUserAccount3::Ancestors
IDSSUserAcctSvcs
IDSSUserAcctSvcs::AddMember
IDSSUserAcctSvcs::Administrator
IDSSUserAcctSvcs::ClearMembership
IDSSUserAcctSvcs::CreateGroup
IDSSUserAcctSvcs::CreateUser
IDSSUserAcctSvcs::CreateUserProfile
IDSSUserAcctSvcs::DeleteGroup
IDSSUserAcctSvcs::DeleteUser
IDSSUserAcctSvcs::DeleteUserProfile
IDSSUserAcctSvcs::DuplicateGroup
IDSSUserAcctSvcs::DuplicateUser
IDSSUserAcctSvcs::Everyone
IDSSUserAcctSvcs::Groups
IDSSUserAcctSvcs::LoadUsers
IDSSUserAcctSvcs::Public
IDSSUserAcctSvcs::RemoveMember
IDSSUserAcctSvcs::SystemAdmins
IDSSUserAcctSvcs::SystemMonitors
IDSSUserAcctSvcs::Users
IDSSUserAcctSvcs2
IDSSUserAcctSvcs2::DeleteAllUserProfile
IDSSUserAcctSvcs2::FindProfileFolder
IDSSUserAcctSvcs2::GetProfileProjects
IDSSUserAcctSvcs2::IsCreateProfile
IDSSUserAcctSvcs2::MergeUsers
IDSSUserAcctSvcs3
IDSSUserAcctSvcs3::CreateProfileFolder
IDSSUserAcctSvcs4
IDSSUserAcctSvcs4::ApplyDeltaXML
IDSSUserAcctSvcs4::get_UserSecurityFilter Mappings
IDSSUserAcctSvcs5
IDSSUserAcctSvcs5::CreateUserEx
IDSSUserAuditor
IDSSUserAuditor::AuditUsers
IDSSUserAuditor::Count
IDSSUserAuditor::Initialize
IDSSUserAuditor::LicensesForCollectionUser
IDSSUserAuditor::LicensesForRole
IDSSUserAuditor::LicensesForUser
IDSSUserAuditor::TotalsCount
IDSSUserAuditor::TotalsLicenseType
IDSSUserAuditor::TotalsReset
IDSSUserAuditor::TotalsValue
IDSSUserAuditor::UsersAuditedCount
IDSSUserAuditor2
IDSSUserAuditor2::CompliantPrivilegesBitMask
IDSSUserAuditor2::IsPrivilegeCompliant
IDSSUserAuditor2::MapLicenseNameToLicense
IDSSUserAuditor2::MapLicensesToPrivileges
IDSSUserAuditor2::MapLicenseToLicenseName
IDSSUserAuditor2::MapPrivilegesToLicenses
IDSSUserAuditor3
IDSSUserAuditor3::GetUsersByLicenseType
IDSSUserAuditor3::LicensesPrivilegeMapXML
IDSSUserFullPrivileges
IDSSUserFullPrivileges::ApplyDeltaXML
IDSSUserFullPrivileges::Count
IDSSUserFullPrivileges::HasNamedPrivilege
IDSSUserFullPrivileges::HasNamedPrivilege2
IDSSUserFullPrivileges::Item
IDSSUserFullPrivileges::Trustee
IDSSUserFullPrivileges::UserRuntime
IDSSUserGoverning
IDSSUserGoverning::AccountType
IDSSUserGoverning::FirstJob
IDSSUserGoverning::MaximumIdleTime
IDSSUserGoverning::MaximumJobs
IDSSUserGoverning::MaximumJobsInQueue
IDSSUserGoverning::MaximumJobTime
IDSSUserGoverning::MaximumResultRows
IDSSUserGoverning::MaximumScheduledTime
IDSSUserGoverning::MaximumSharedLogins
IDSSUserRuntime
IDSSUserRuntime::AccessToken
IDSSUsers
IDSSUsers::_NewEnum
IDSSUsers::Add
IDSSUsers::Clear
IDSSUsers::Count
IDSSUsers::Item
IDSSUsers::Remove
IDSSUserSecurityFilterMappings
IDSSUserSecurityFilterMappings::Add
IDSSUserSecurityFilterMappings::Count
IDSSUserSecurityFilterMappings::IncludeGroup
IDSSUserSecurityFilterMappings::Looksup
IDSSUserSecurityFilterMappings::MapProject
IDSSUserSecurityFilterMappings:: MapSecurityFilter
IDSSUserSecurityFilterMappings::MapUser
IDSSUserSecurityFilterMappings::Project
IDSSUserSecurityFilterMappings::Remove
IDSSUserSecurityFilterMappings::User
IDSSUserSecurityFilters
IDSSUserSecurityFilters::_NewEnum
IDSSUserSecurityFilters::Add
IDSSUserSecurityFilters::Clear
IDSSUserSecurityFilters::Count
IDSSUserSecurityFilters::Item
IDSSUserSecurityFilters::Remove
IDSSUserSession
IDSSUserSession::ClientMachineName
IDSSUserSession::ClientType
IDSSUserSession::FinishedJobCount
IDSSUserSession::InfoType
IDSSUserSession::Name
IDSSUserSession::NetAddress
IDSSUserSession::ProjectLoginTo
IDSSUserSession::ServerContextID
IDSSUserSession::SessionID
IDSSUserSession::TimeConnected
IDSSUserSession::TimeFirstJob
IDSSUserSession::TimeLastJob
IDSSUserSession::TotalJobCount
IDSSUserSessions
IDSSUserSessions::_NewEnum
IDSSUserSessions::Count
IDSSUserSessions::Item
IDSSUserSessions::ItemById
IDSSUses
IDSSUses::_NewEnum
IDSSUses::Add

IDSSUses::Clear
IDSSUses::Count
IDSSUses::Item
IDSSUses::NotType
IDSSUses::Remove
IDSSUses::Type
IDSSValidationErrors
IDSSValidationIssue
IDSSValidationIssue::CauseFormID
IDSSValidationIssue::CauseObjectID
IDSSValidationIssue::CauseObjectType
IDSSValidationIssue::IsRecoverable
IDSSValidationIssue::IssueType
IDSSValidationIssue::LocationObjectID
IDSSValidationIssue::LocationObjectType
IDSSValidationIssue::LocationThreshold
IDSSValidationIssue::LocationType
IDSSValidationIssues
IDSSValidationIssues::_NewEnum
IDSSValidationIssues::Count
IDSSValidationIssues::IsRecoverable
IDSSValidationIssues::Item
IDSSWebServerAdmin
IDSSWebServerAdmin::Close
IDSSWebServerAdmin::ConnectionPoolSize
IDSSWebServerAdmin::DSSServerName
IDSSWebServerAdmin::Open
IDSSWebServerAdmin::ServerNet
IDSSWebServerAdmin::SharedConnections
IDSSWebServerAdmin::Status
IDSSWebServerSharedConnection
IDSSWebServerSharedConnection::Close
IDSSWebServerSharedConnection::NetAddress
IDSSWebServerSharedConnection::Status
IDSSWebServerSharedConnection::UserName
IDSSWebServerSharedConnections
IDSSWebServerSharedConnections::_NewEnum
IDSSWebServerSharedConnections::Count
IDSSWebServerSharedConnections::Item
IDSSWorkingSet
IDSSWorkingSet::AttributeForms
IDSSWorkingSet::Definition
IDSSWorkingSet::Filter
IDSSWorkingSet::GetElements
IDSSWorkingSet::GetElementsXML
IDSSWorkingSet::IsDerived
IDSSWorkingSet::ObjectAlias
IDSSWorkingSet::Template
IDSSWorkingSet::TypedUnitObjects
IDSSWorkingSet::UnitObjects
IDSSXMLCompression
IDSSXMLCompression::ApplyDeltaXML
IDSSXMLCompression::Finish
IDSSXMLContext
IDSSXMLContext::AddElementToNode
IDSSXMLContext::AddElementToParent
IDSSXMLContext::AddTextToNode
IDSSXMLContext::AddTextToParent
IDSSXMLContext::AppendToInfo
IDSSXMLContext::AppendToNode
IDSSXMLContext::AppendToParent
IDSSXMLContext::CreateElement
IDSSXMLContext::DeltaPropXml
IDSSXMLContext::DeltaXml
IDSSXMLContext::Document
IDSSXMLContext::FindChild
IDSSXMLContext::GetAttribute
IDSSXMLContext::GraphHeight
IDSSXMLContext::GraphMaxCols
IDSSXMLContext::GraphMaxRows
IDSSXMLContext::GraphStartCol
IDSSXMLContext::GraphStartRow
IDSSXMLContext::GraphType
IDSSXMLContext::GraphWidth
IDSSXMLContext::GridStyle
IDSSXMLContext::InfoNode
IDSSXMLContext::Init
IDSSXMLContext::LastNode
IDSSXMLContext::Level
IDSSXMLContext::Locale
IDSSXMLContext::LookupObject
IDSSXMLContext::MaxCols
IDSSXMLContext::MaxRows
IDSSXMLContext::MessageID
IDSSXMLContext::ParentNode
IDSSXMLContext::RefID
IDSSXMLContext::ResultFlags
IDSSXMLContext::SetAttribute
IDSSXMLContext::StartCol
IDSSXMLContext::StartRow
IDSSXMLContext::StyleXsl
IDSSXMLContext::UserRuntime
IDSSXMLContext::ViewIndex
IDSSXMLContext2
IDSSXMLContext2::IDispatchProperty
IDSSXMLContext2::LongProperty
IDSSXTabDataAccess
IDSSXTabDataAccess::Cell
IDSSXTabDataAccess::CellFlag
IDSSXTabDataAccess::Column
IDSSXTabDataAccess::Data
IDSSXTabDataAccess::NumColumns
IDSSXTabDataAccess::NumRows
IDSSXTabDataAccess::Row
IDSSXTabDataAccess2
IDSSXTabDataAccess2::Cell2
IDSSXTabDataAccess2::Column2
IDSSXTabDataAccess2::Data2
IDSSXTabDataAccess2::Row2
IDSSXTabFactory
IDSSXTabView
IDSSXTabView::Body
IDSSXTabViewAccess
IDSSXTabViewAccess::Iterator
IDSSXTabViewAccess::Metric
IDSSXTabViewAccess::Threshold

MICROSTRATEGY NARROWCAST SERVER API

Delivery Manager

IMSTRDeliveryEngine::CancelExecution
IMSTRDeliveryEngine::GetSystemState
IMSTRDeliveryEngine::ConnectToRepository
IMSTRDeliveryEngine::NewObject
IMSTRDeliveryEngine::ConnectToSystem
IMSTRDeliveryEngine::PauseExecution
IMSTRDeliveryEngine::CopyObject
IMSTRDeliveryEngine::ResumeExecution
IMSTRDeliveryEngine::DeleteObject
IMSTRDeliveryEngine::Root
IMSTRDeliveryEngine::
DisconnectFromRepository
IMSTRDeliveryEngine::SaveObject
IMSTRDeliveryEngine::DisconnectFromSystem
IMSTRDeliveryEngine::SetExecutionEngineOrder
IMSTRDeliveryEngine::ExecuteService
IMSTRDeliveryEngine::SetDistribution
ManagerOrder
IMSTRDeliveryEngine::FindObject
IMSTRDeliveryEngine::StartDistributionManager
IMSTRDeliveryEngine::GetExecutionEngineState
IMSTRDeliveryEngine::StartSystem
IMSTRDeliveryEngine::GetDistribution
ManagerState
IMSTRDeliveryEngine::StopDistributionManager
IMSTRDeliveryEngine::GetExecutionQueue
IMSTRDeliveryEngine::StopSystem
IMSTRDeliveryEngine::GetObject
IMSTRDeliveryEngine::SystemLocale
IMSTRDeliveryEngine::HiddenObjectsFolder
IMSTRDeliveryEngine::NewObjectFromXML
IMSTRDeliveryEngine::CheckValidAction
IMSTRDeliveryEngine::SystemID
IMSTRAccountDefinition
IMSTRInformationSourcePool
IMSTRAlias
IMSTRInformationSourcePoolNotify
IMSTRAliases
IMSTRInformationTransmissionModule
Definition
IMSTRClusterDefinition
IMSTRLocaleDefinition
IMSTRDeliveryEngineDefinition
IMSTRLocalizedDescriptionDefinition
IMSTRDeviceDefinition
IMSTRLocalizedDescriptions
IMSTRDistributionManagerDefinition
IMSTRLocalizedNameDefinition
IMSTRDistributionSetDefinition
IMSTRLocalizedNames
IMSTRDistributionSets
IMSTRObjectInfoDefinition
IMSTRDocumentDefinition
IMSTRObjectInfos
IMSTRDocumentElementDefinition
IMSTRObjectPaths
IMSTRDocumentElements
IMSTRObjectPropertyStrings
IMSTRDocumentFormattingModuleDefinition
IMSTRObjectSubTypes IMSTRDocument
FormattingPropertySetDefinition
IMSTRObjectTypes
IMSTRDocumentSectionSetDefinition
IMSTRPluginDialog
IMSTRDocumentSectionSets
IMSTRPublicationDefinition
IMSTREmbeddedObjects
IMSTRQuestionObjectDefinition
IMSTRErrorInfoDefinition
IMSTRScheduleDefinition
IMSTRFileDefinition
IMSTRSearchDefinition
IMSTRFolderDefinition
IMSTRServiceDefinition
IMSTRFormattedContentDefinition
IMSTRServiceExecutionSegmentProperties
Definition
IMSTRInformationObjectDefinition
IMSTRSubscriptionBookModuleDefinition
IMSTRInformationSourceDefinition
IMSTRSubscriptionSetDefinition
IMSTRInformationSourceModuleDefinition
IMSTRInformationSource::
OpenInformationSource
IMSTRInformationSource::
LoadServiceDefinition
IMSTRInformationSource::
ExecuteInformationObject
IMSTRInformationSource::ExecuteSubscription
InformationObject
IMSTRInformationSource::ExecuteSubscription
SegmentSet
IMSTRInformationSource::
ReleaseServiceExecution
IMSTRInformationSource::
UnloadServiceDefinition
IMSTRInformationSource::
CloseInformationSource

IMSTRInformationSource::PauseServiceExecution
IMSTRInformationSource::ResumeServiceExecution
IMSTRInformationSource::CancelServiceExecution
IMSTRInformationSourceNotify::TransmitCallback
IMSTRObjectBrowser::OpenInformationSourceObjectBrowser
IMSTRObjectBrowser::CloseInformationSourceObjectBrowser
IMSTRObjectBrowser::SetInformationSourceLocaleProps
IMSTRObjectBrowser::DefineInformationObject
IMSTRObjectBrowser::DefineQuestionObject
IMSTRObjectBrowser::DefineAuthenticationObject
IMSTRObjectBrowser::DefineSecurityObject
IMSTRObjectBrowser::DefinePreferenceObject
IMSTRInformationSourceModule3::Name
IMSTRInformationSourceModule3::Description
IMSTRInformationSourceModule3::InformationSourceProgID
IMSTRInformationSourceModule3::SetInformationSourceProperties
IMSTRInformationSourceModule3::InformationSourceModuleType
IMSTRInformationSourceModule3::QuestionObjectDefinitionFormat
IMSTRInformationSourceModule3::GetAliasInformation
IMSTRInformationSourceModule3::RemoveInformationSource
IMSTRInformationTransmissionModule::Name
IMSTRInformationTransmissionModule::Description
IMSTRInformationTransmissionModule::TransmitterProgID
IMSTRInformationTransmissionModule::SectionNames
IMSTRInformationTransmissionModule::SectionRules
IMSTRInformationTransmissionModule::GetConfigDialogForTransmitter
IMSTRInformationTransmissionModule::GetConfigDialogForDevice
IMSTRInformationTransmissionModule::GetConfigDialogForLocale
IMSTRInformationTransmissionModule::GetConfigDialogForSubscription
IMSTRInformationTransmitter::Initialize
IMSTRInformationTransmitter::Transmit
IMSTRInformationTransmitter::Pause
IMSTRInformationTransmitter::Continue
IMSTRInformationTransmitter::Cancel
IMSTRInformationTransmitter::Terminate
IMSTRInformationTransmitterNotify::TransmitCallback
IMSTRDocumentFormattingModule::Name
IMSTRDocumentFormattingModule::Description
IMSTRDocumentFormattingModule::FormatterProgID
IMSTRDocumentFormattingModule::SupportedDocumentType
IMSTRDocumentFormattingModule::GetConfigDialogForDevice
IMSTRDocumentFormattingModule::GetConfigDialogForFormatter
IMSTRDocumentFormattingModule::GetConfigDialogForLocale
IMSTRDocumentFormatter::Initialize
IMSTRDocumentFormatter::FormatDocumentElement
IMSTRDocumentFormatter::FormatInsertionString
IMSTRDocumentFormatter::FormatDocument
IMSTRDocumentFormatter::Terminate
IMSTRSubscriptionBookModule::Name
IMSTRSubscriptionBookModule::Description
IMSTRSubscriptionBookModule:InformationSourceProgID
IMSTRSubscriptionBookModule:GetConfigurationDialog
IMSTRSubscriptionBookObjectAdministrator2::OpenSubscriptionBook ObjectAdministrator
IMSTRSubscriptionBookObjectAdministrator2::CloseSubscriptionBook ObjectAdministrator
IMSTRSubscriptionBookObjectAdministrator2::ManageUserBook
IMSTRSubscriptionBookObjectAdministrator2::DefineSubscriptionSet
IMSTRSubscriptionBookObjectAdministrator2::IsObjectUsed
IMSTRSubscriptionBookObjectAdministrator2::InvalidateObject
IMSTRSubscriptionBookObjectAdministrator2::DeleteDepdendents
IMSTRSubscriptionBookObjectAdministrator2::Copy SubscriptionSet
IMSTRSubscriptionBookObjectAdministrator2::Editor Container
IMSTRSubscriptionBook::OpenSubscriptionBook
IMSTRSubscriptionBook::CloseSubscriptionBook
IMSTRSubscriptionBook::SegmentSubscriptionSet
IMSTRSubscriptionBook::GetSubscriptionSet
IMSTRSubscriptionBook::ReleaseSegments
IMSTRSubscriptionBook::InvalidateObject
IConsumerObject::Connect
IConsumerObject::RegisterCallback
IConsumerObject::RegisterCallback2
IConsumerObject::Subscribe
IConsumerObject::Disconnect

Subscription Portal

IAddress::NODE_ADD_DISPLAY
IAddress::NODE_ADDRESS_ID
IAddress::NODE_ADDRESS_NAME
IAddress::NODE_DELIVERY_WINDOW
IAddress::NODE_DEVICE_ID
IAddress::NODE_PHYSICAL_ADD
IAddress::NODE_PIN
IAddress::NODE_STATUS
IAddress::NODE_TIMEZONE_ID
IAddress::NODE_TRANSPROPS_ID
IAddress::POS_ADD_DISPLAY
IAddress::POS_ADDRESS_ID
IAddress::POS_ADDRESS_NAME
IAddress::POS_CREATED_BY
IAddress::POS_CREATION_DATE
IAddress::POS_DELIVERY_WINDOW
IAddress::POS_DEVICE_ID
IAddress::POS_EXPIRATION_DATE
IAddress::POS_LAST_MOD_BY
IAddress::POS_LAST_MOD_DATE
IAddress::POS_PHYSICAL_ADD
IAddress::POS_PIN
IAddress::POS_STATUS
IAddress::POS_TIMEZONE_ID
IAddress::POS_TRANSPROPS_ID
IAddress::SIZE_ADDRESS_PROPS
IAddress::addAddress
IAddress::deleteAddresses
IAddress::editAddress
IAddress::getAddressInfo
IAddress::getUserAddresses
IAddress::getUserAddressesForDevice
IAddress::getUserAddressesForService
IAddress::getUserAddressesForServices
IAddress::getUserAddressInfoForServices
IAdmin::LOGGING_LEVEL_HIGH
IAdmin::LOGGING_LEVEL_LOW
IAdmin::addVirtualDirectory
IAdmin::checkDBAlias
IAdmin::connectToSubscriptionEngines
IAdmin::createDBAlias
IAdmin::deleteDBAlias
IAdmin::deleteVirtualDirectory
IAdmin::generateSubscriptionSetSQL
IAdmin::getColumns
IAdmin::getDBAliases
IAdmin::getInstallationPath
IAdmin::getLoggingLevel
IAdmin::getMetadataConnectionProperties
IAdmin::getSiteConfigurationProperties
IAdmin::getSubscriptionEngineLocation
IAdmin::getTables
IAdmin::getVirtualDirectories
IAdmin::resetSubscriptionEngines
IAdmin::saveMetadataConnectionProperties
IAdmin::saveSiteConfigurationProperties
IAdmin::saveSubscriptionEngineLocation
IAdmin::setLoggingLevel
IAdmin::updateDBAlias
IEnterpriseRepository::CHUNK_SIZE
IEnterpriseRepository::KEY_DATA
IEnterpriseRepository::KEY_DOCUMENT_ID
IEnterpriseRepository::KEY_EXECUTION_TIME
IEnterpriseRepository::KEY_EXPIRATION_TIME
IEnterpriseRepository::KEY_INDEX
IEnterpriseRepository::cleanExpiredServices
IEnterpriseRepository::expireDocument
IEnterpriseRepository::expireSubscriptions

IEnterpriseRepository::getAttachmentsList
IEnterpriseRepository::getAttachmentType
IEnterpriseRepository::getAvailableSubscriptions
IEnterpriseRepository::getDocumentID
IEnterpriseRepository::getDocumentInfo
IEnterpriseRepository::getSubscriptionInfo
IEnterpriseRepository::getSubscriptionsList
IEnterpriseRepository::readAttachment
IEnterpriseRepository::readAttachmentBlock
IEnterpriseRepository::readDocument
IEnterpriseRepository::readSubscription
IEnterpriseRepository::writeAttachment
IEnterpriseRepository::writeDocument
IPersonalization::EDIT_PROFILE_DESC
IPersonalization::EDIT_PROFILE_NAME
IPersonalization::KEY_PREFERENCE_DESCRIPTION
IPersonalization::KEY_PREFERENCE_NAME
IPersonalization::KEY_PROFILE_DESCRIPTION
IPersonalization::KEY_PROFILE_NAME
IPersonalization::KEY_PUBLICATION_ID
IPersonalization::createProfile
IPersonalization::deleteProfile
IPersonalization::deleteSubscriptionPersonalization
IPersonalization::getDetailsForQuestions
IPersonalization::getPreferenceObjects
IPersonalization::getProfile
IPersonalization::getProfileNamesForQuestions
IPersonalization::getQuestionsAndProfilesForPublication
IPersonalization::getQuestionsAndProfilesForSubscriptionSet
IPersonalization::getQuestionsForPublication
IPersonalization::getSubscriptionPersonalization
IPersonalization::getUserDefaultPersonalization
IPersonalization::saveDefaultSubscriptionPersonalization
IPersonalization::savePreferenceObjects
IPersonalization::saveSubscriptionPersonalization
IPersonalization::updatePreferenceObjects
IPersonalization::updateProfile
ISiteInfo::KEY_DEVICE_TYPE_ID
ISiteInfo::ROOT_APP_FOLDER_TYPE
ISiteInfo::ROOT_DEVICE_FOLDER_TYPE
ISiteInfo::TYPE_ADDRESS
ISiteInfo::TYPE_COLUMN
ISiteInfo::TYPE_DBALIAS
ISiteInfo::TYPE_LOGGINGLEVEL
ISiteInfo::TYPE_MAPPING
ISiteInfo::TYPE_PREFERENCEOBJECT
ISiteInfo::TYPE_PROFILE
ISiteInfo::TYPE_QUESTION
ISiteInfo::TYPE_SERVICE
ISiteInfo::TYPE_SUBSCRIPTION_SET
ISiteInfo::TYPE_TABLE
ISiteInfo::createChannel
ISiteInfo::createDeviceType
ISiteInfo::createDeviceTypeDefinitions
ISiteInfo::createMappingDefinition
ISiteInfo::createMetadataTables
ISiteInfo::createObject
ISiteInfo::createSite
ISiteInfo::createTransmissionProperties
ISiteInfo::deleteChannel
ISiteInfo::deleteChannelProperties
ISiteInfo::deleteDeviceType
ISiteInfo::deleteDeviceTypeDefinitions
ISiteInfo::deleteMappingDefinition
ISiteInfo::deleteObject
ISiteInfo::deleteObjectProperties
ISiteInfo::deleteSite
ISiteInfo::deleteSiteProperties
ISiteInfo::deleteTransmissionProperties
ISiteInfo::getAllInformationSources
ISiteInfo::getAllLocales
ISiteInfo::getAllSites
ISiteInfo::getChannelProperties
ISiteInfo::getChannelsForSite
ISiteInfo::getDetailsForQuestions
ISiteInfo::getDeviceTypeDefinitions
ISiteInfo::getDeviceTypes
ISiteInfo::getFolderContents
ISiteInfo::getInformationSourcesForSite
ISiteInfo::getMappingDefinition
ISiteInfo::getMappingObjects
ISiteInfo::getNamedSchedulesForService
ISiteInfo::getObjectParentInfo
ISiteInfo::getObjectProperties
ISiteInfo::getObjectsForSite
ISiteInfo::getQuestionsForService
ISiteInfo::getSiteProperties
ISiteInfo::getSubscriptionSetsForService
ISiteInfo::getTransmissionProperties
ISiteInfo::insertChannelProperties
ISiteInfo::insertObjectProperties
ISiteInfo::insertSiteProperties
ISiteInfo::updateChannelProperties
ISiteInfo::updateDeviceTypeDefinitions
ISiteInfo::updateDeviceTypeProperties
ISiteInfo::updateObjectProperties
ISiteInfo::updateSiteProperties
ISiteInfo::updateTransmissionProperties
ISubscription::ARRAY_SIZE
ISubscription::KEY_GUID
ISubscription::POS_ACCOUNT_ID
ISubscription::POS_ADD_TRANS_PROPS
ISubscription::POS_ADDRESS_ID
ISubscription::POS_CHANNEL_ID
ISubscription::POS_CREATED_BY
ISubscription::POS_CREATED_DATE
ISubscription::POS_EXPIRATION_DATE
ISubscription::POS_LAST_MOD_BY
ISubscription::POS_LAST_MOD_DATE
ISubscription::POS_STATUS
ISubscription::POS_SUBSCRIPTION_GUID
ISubscription::POS_SUBSCRIPTION_ID
ISubscription::POS_SUBSCRIPTION_SET_ID
ISubscription::POS_TRANS_PROPS_ID
ISubscription::addSubscription
ISubscription::deleteSubscriptions
ISubscription::editSubscription
ISubscription::getDeliveryOptions
ISubscription::getSubscription
ISubscription::getUserSubscriptions
ISubscription::getUserSubscriptionSets
ISubscriptionFactory::createAddress()
ISubscriptionFactory::createAdmin()
ISubscriptionFactory::createEnterpriseRepository()
ISubscriptionFactory::createPersonalization()
ISubscriptionFactory::createSiteInfo()
ISubscriptionFactory::createSubscription()
ISubscriptionFactory::createSystemInfo()
ISubscriptionFactory::createUser()
ISystemInfo::KEY_FOLDER_ID
ISystemInfo::KEY_INFO_SOURCE_ID
ISystemInfo::KEY_INFO_SOURCE_MODULE_ID
ISystemInfo::KEY_OBJECT_ID
ISystemInfo::KEY_PRODUCT_NAME
ISystemInfo::KEY_VERSION
ISystemInfo::VALUE_PRODUCT_NAME
ISystemInfo::VERSION_NUMBER
ISystemInfo::checkCompatibility
ISystemInfo::getAllLocales
ISystemInfo::getDevicesForService
ISystemInfo::getDevicesInFolders
ISystemInfo::getFolderContents
ISystemInfo::getInformationSourceDefinition
ISystemInfo::getInformationSourceModuleDefinition
ISystemInfo::getNamedSchedulesForService
ISystemInfo::getObjectInfo
ISystemInfo::getVersion
IUser::KEY_AGREEMENT_ID
IUser::KEY_DEF_ADD_ID
IUser::KEY_HINT
IUser::KEY_STATUS
IUser::KEY_USER_NAME
IUser::KEY_USER_PASSWORD
IUser::KEY_USER_UPNAME
IUser::POS_ACCOUNT_ID
IUser::POS_AGREEMENT_ID
IUser::POS_CREATED_BY
IUser::POS_CREATED_DATE
IUser::POS_DEF_ADD_ID
IUser::POS_EXPIRATION_DATE
IUser::POS_HINT
IUser::POS_LAST_MOD_BY
IUser::POS_LAST_MOD_DATE
IUser::POS_LOCALE_ID
IUser::POS_STATUS
IUser::POS_USER_ID
IUser::POS_USER_NAME
IUser::POS_USER_PASSWORD
IUser::USER_ARRAY_SIZE
IUser::closeSession
IUser::createSession
IUser::createUser
IUser::deactivateUser
IUser::deleteUser
IUser::getUserAuthenticationObjects
IUser::getUserHint
IUser::getUserProperties
IUser::getUserSecurityObjects
IUser::isSessionValid
IUser::saveUserAuthenticationObjects
IUser::saveUserSecurityObjects
IUser::updateUserAuthenticationObjects
IUser::updateUserDefaultAddress
IUser::updateUserHint
IUser::updateUserLocale
IUser::updateUserPassword
IUser::updateUserProperties
IUser::updateUserSecurityObjects

COPYRIGHT NOTICE Copyright ©2005 MicroStrategy Incorporated, 1861 International Drive, McLean, Virginia, 22102 U.S.A. All rights reserved.

The material presented herein is based on information we consider reliable. We do not represent that it is accurate and complete. No person should consider our distribution of this material as making any representation or warranty with respect to such material and should not rely upon it as such.

This product is patented. One or more of the following patents may apply to the product sold herein: U.S. Patent Nos. 6,154,766, 6,173,310, 6,260,050, 6,263,051, 6,269,393, 6,279,033, 6,501,832, 6,567,796, 6,587,547, 6,606,596, 6,658,093, 6,658,432, 6,662,195, 6,671,715, 6,691,100, 6,694,316, 6,697,808, 6,704,723, 6,707,889, 6,741,980, 6,765,997, 6,768,788, 6,772,137, 6,788,768, 6,792,086, 6,798,867, 6,801,910, 6,820,073, 6,8 e pending.

TRADEMARKS: MICROSTRATEGY, MICROSTRATEGY 6, MICROSTRATEGY 7, MICROSTRATEGY 7i, MICROSTRATEGY 7i EVALUATION EDITION, MICROSTRATEGY 7i OLAP SERVICES, MICROSTRATEGY 8, MICROSTRATEGY EVALUATION EDITION, MICROSTRATEGY ADMINISTRATOR, MICROSTRATEGY AGENT, MICROSTRATEGY ARCHITECT, MICROSTRATEGY BI DEVELOPER KIT, MICROSTRATEGY BROADCAST SERVER, MICROSTRATEGY BROADCASTER, MICROSTRATEGY BROADCASTER SERVER, MICROSTRATEGY BUSINESS INTELLIGENCE PLATFORM, MICROSTRATEGY CONSULTING, MICROSTRATEGY CRM APPLICATIONS, MICROSTRATEGY CUSTOMER ANALYZER, MICROSTRATEGY DESKTOP, MICROSTRATEGY DESKTOP ANALYST, MICROSTRATEGY DESKTOP DESIGNER, MICROSTRATEGY ECRM 7, MICROSTRATEGY EDUCATION, MICROSTRATEGY ETRAINER, MICROSTRATEGY EXECUTIVE, MICROSTRATEGY INFOCENTER, MICROSTRATEGY INTELLIGENCE SERVER, MICROSTRATEGY INTELLIGENCE SERVER UNIVERSAL EDITION, MICROSTRATEGY MDX ADAPTER, MICROSTRATEGY NARROWCAST SERVER, MICROSTRATEGY OBJECTS, MICROSTRATEGY OLAP PROVIDER, MICROSTRATEGY SDK, MICROSTRATEGY SUPPORT, MICROSTRATEGY TELECASTER, MICROSTRATEGY TRANSACTOR, MICROSTRATEGY WEB, MICROSTRATEGY WEB BUSINESS ANALYZER, MICROSTRATEGY WORLD, ALARM, ALARM.COM, ALERT.COM, ANGEL, ANGEL.COM, APPLICATION DEVELOPMENT AND SOPHISTICATED ANALYSIS, BEST IN BUSINESS INTELLIGENCE, CENTRALIZED APPLICATION MANAGEMENT, CHANGING THE WAY GOVERNMENT LOOKS AT INFORMATION, DSS AGENT, DSS ARCHITECT, DSS BROADCASTER, DSS BROADCASTER SERVER, DSS OFFICE, DSS SERVER, DSS SUBSCRIBER, DSS TELECASTER, DSS WEB, EBROADCASTER, ECASTER, ESTRATEGY, ETELECASTER, INFORMATION LIKE WATER, INSIGHT IS EVERYTHING, INTELLIGENCE THROUGH EVERY PHONE, INTELLIGENCE TO EVERY DECISION MAKER, INTELLIGENT E-BUSINESS, IWAPU, PERSONAL INTELLIGENCE NETWORK, PERSONALIZED INTELLIGENCE PORTAL, QUERY TONE, QUICKSTRIKE, RAPID APPLICATION DEVELOPMENT, STRATEGY.COM, TELEPATH, TELEPATH INTELLIGENCE, TELEPATH INTELLIGENCE (AND DESIGN), THE E-BUSINESS INTELLIGENCE PLATFORM, THE FOUNDATION FOR INTELLIGENT E-BUSINESS, THE INTEGRATED BUSINESS INTELLIGENCE PLATFORM BUILT FOR THE ENTERPRISE, THE INTELLIGENCE COMPANY, THE PLATFORM FOR INTELLIGENT E-BUSINESS, THE POWER OF INTELLIGENT EBUSINESS, THE POWER OF INTELLIGENT E-BUSINESS, THE SCALABLE BUSINESS INTELLIGENCE PLATFORM BUILT FOR THE INTERNET, INDUSTRIAL-STRENGTH BUSINESS INTELLIGENCE, AND OFFICE INTELLIGENCE ARE ALL REGISTERED TRADEMARKS OR TRADEMARKS OF MICROSTRATEGY INCORPORATED. ALL OTHER PRODUCTS ARE TRADEMARKS OF THEIR RESPECTIVE HOLDERS. SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE. MICROSTRATEGY IS NOT RESPONSIBLE FOR ERRORS OR OMISSIONS. MICROSTRATEGY MAKES NO WARRANTIES OR COMMITMENTS CONCERNING THE AVAILABILITY OF FUTURE PRODUCTS OR VERSIONS THAT MAY BE PLANNED OR UNDER DEVELOPMENT.

MicroStrategy®
Best In Business Intelligence™

COLL-0596 1105

MicroStrategy Incorporated • 1861 International Drive McLean, VA 22102 • 703.848.8600 • www.microstrategy.com