MicroStrategy

Analytics and Mobility

MDX Cube Reporting Guide

MicroStrategy ONE

MicroStrategy ONE

April 2024

Copyright © 2024 by MicroStrategy Incorporated. All rights reserved.

Trademark Information

The following are either trademarks or registered trademarks of MicroStrategy Incorporated or its affiliates in the United States and certain other countries:

Dossier, Enterprise Semantic Graph, Expert.Now, Hyper.Now, HyperIntelligence, HyperMobile, HyperVision, HyperWeb, Intelligent Enterprise, MicroStrategy, MicroStrategy 2019, MicroStrategy 2020, MicroStrategy 2021, MicroStrategy AI, MicroStrategy Analyst Pass, MicroStrategy Architect, MicroStrategy Architect Pass, MicroStrategy Auto, MicroStrategy Cloud, MicroStrategy Cloud Intelligence, MicroStrategy Command Manager, MicroStrategy Communicator, MicroStrategy Consulting, MicroStrategy Desktop, MicroStrategy Developer, MicroStrategy Distribution Services, MicroStrategy Education, MicroStrategy Embedded Intelligence, MicroStrategy Enterprise Manager, MicroStrategy Federated Analytics, MicroStrategy Geospatial Services, MicroStrategy Identity, MicroStrategy Identity Manager, MicroStrategy Identity Server, MicroStrategy Insights, MicroStrategy Integrity Manager, MicroStrategy Intelligence Server, MicroStrategy Library, MicroStrategy Mobile, MicroStrategy Narrowcast Server, MicroStrategy ONE, MicroStrategy Object Manager, MicroStrategy Office, MicroStrategy OLAP Services, MicroStrategy Parallel Relational In-Memory Engine (MicroStrategy PRIME), MicroStrategy R Integration, MicroStrategy Report Services, MicroStrategy SDK, MicroStrategy System Manager, MicroStrategy Transaction Services, MicroStrategy Usher, MicroStrategy Web, MicroStrategy Workstation, MicroStrategy World, Usher, and Zero-Click Intelligence.

The following design marks are either trademarks or registered trademarks of MicroStrategy Incorporated or its affiliates in the United States and certain other countries:

Other product and company names mentioned herein may be the trademarks of their respective owners.

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.

CONTENTS

About MDX Cube Sources in MicroStrategy

Many companies have stored data that is specifically structured for query, reporting, and analysis, known as a data warehouse, and they also have data stored in an MDX cube source such as SAP Business Intelligence Warehouse (SAP BW), Microsoft Analysis Services (Analysis Services), Oracle Essbase, or IBM Cognos TM1 (TM1). This system setup requires an integrated business intelligence system (BI), such as MicroStrategy, that can concurrently access both the MDX cube source and the data warehouse effectively.

MicroStrategy provides a rich set of functionalities ranging from OLAP Services and Report Services to Narrowcast capabilities, all of which can be exposed in a unified Web interface. Using the MicroStrategy standard interface, Intelligence Server can join data from different MDX cube sources, in addition to relational databases, and bring the data into one cohesive user community known as a MicroStrategy project. These additional MDX cube sources include the following:

- SAP BW 7.3, and 7.4
- Microsoft Analysis Services 2012, 2014, 2016, 2017, 2019, and Azure
- You can also use Microsoft Analysis Services 2012, 2014, 2016, 2017, 2019, and Azure in Tabular Mode using InMemory as the query mode.
- For purposes of integrating with MicroStrategy, these versions of Microsoft Analysis Services provide the same features and functionality.
- Oracle Essbase Server 11.1.2
- IBM Cognos TM1 10.1.1 and 10.2

MicroStrategy integration with MDX cube sources does not change the overall structure of MicroStrategy software. Rather, integration allows MicroStrategy to gain additional data sources for analysis, treating each as a data source to be used in MicroStrategy for query, reporting, and analysis.

This section builds on basic knowledge of MicroStrategy and MDX cube source terminology and architecture. It describes how MicroStrategy

Intelligence Server converts MDX cube source data and objects into MicroStrategy using MultiDimensional Expressions (MDX). SAP BW obtains data from R/3, CRM, SEM, or another SAP data source system. This data is stored in cubes or other SAP objects. Likewise, Analysis Services, Oracle Essbase, and TM1 store data in cubes obtained from various sources. To access the data and convert it into MicroStrategy standard objects and conventions, Intelligence Server generates MDX. Defined by Microsoft, MDX is similar to SQL but is used to query cubes. An MDX expression returns a multidimensional result set (dataset) that consists of axis data, cell data, and properties data. For more information on MDX syntax, visit *http://msdn.microsoft.com/* and search for MDX.

If you use MDX cube sources and MicroStrategy as your combined BI solution, you can get the best out of both, including the following:

- Access to MDX cube sources and a regular data warehouse
- Five styles of BI
- Custom development of reports and applications
- Transaction-level analysis
- Integration with other systems via Web Services

If you upgrade a pre-9.0 MicroStrategy project that includes MDX cubes, see the Upgrade Help for information on updating MDX objects to enhance the performance of using these objects in MicroStrategy.

For troubleshooting and diagnostics logging routines related to MDX cube sources, see the *Troubleshooting* section of the System Administration Help.

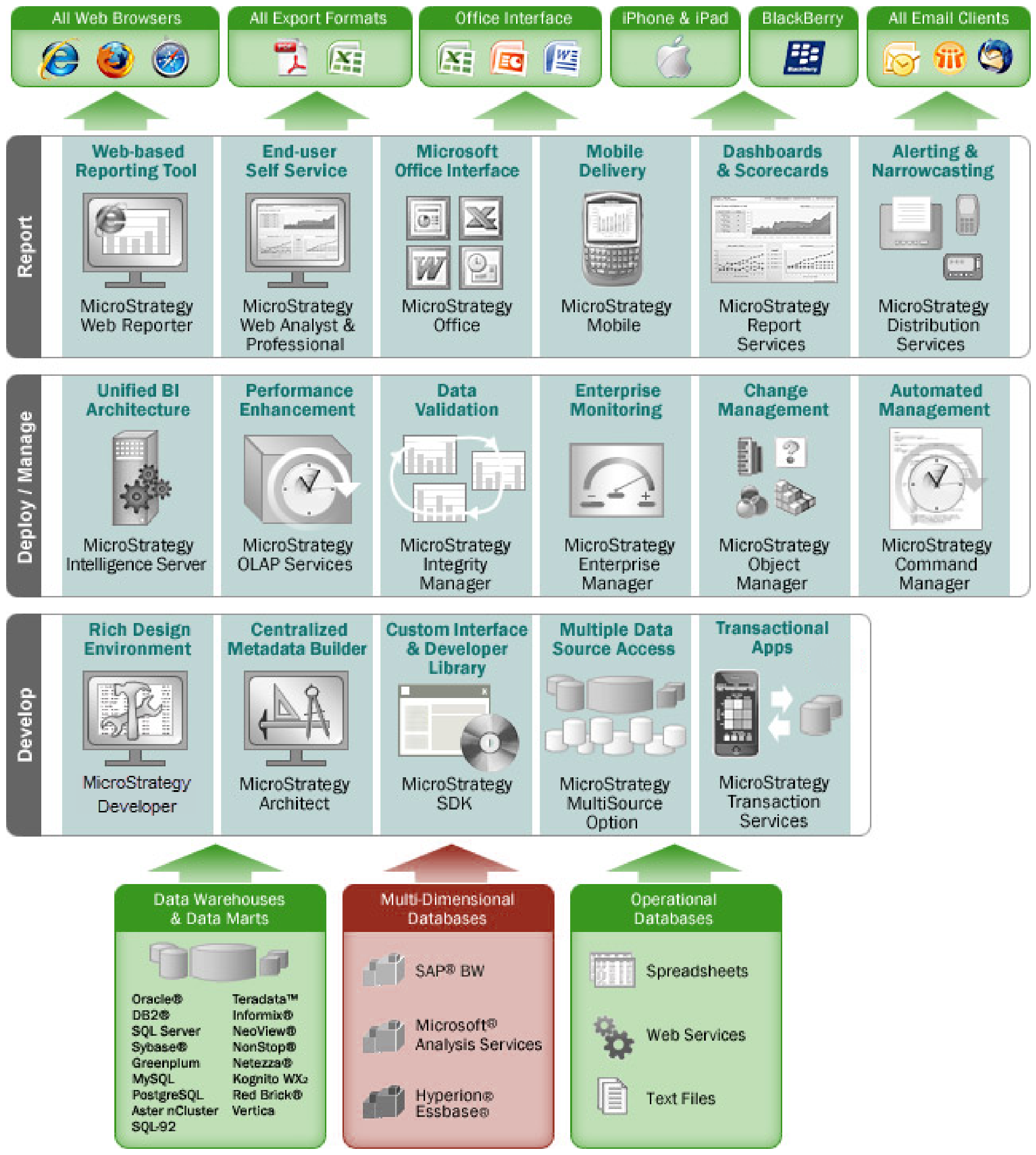
Understanding MicroStrategy Architecture

The MicroStrategy platform offers OLAP Services, Report Services, and Narrowcast Server functionality, all of which can be accessed through MicroStrategy Web. Support for SAP BW, Analysis Services, Oracle Essbase, TM1, MicroStrategy Freeform SQL, and MicroStrategy Query

Builder provides additional mechanisms for pulling data into the MicroStrategy platform for analysis, as illustrated in the diagram below.

For information on Freeform SQL and Query Builder reporting, see the Advanced Reporting Help.

Data is pulled from multiple MDX cube sources using MDX, and from operational systems using Freeform SQL or Query Builder. Once the data is retrieved, it is treated in the same manner as data pulled from a relational data warehouse. This means that core MicroStrategy reporting capabilities are available no matter what the original data source is.



MicroStrategy uses a repository known as a metadata, whose data associates the tables and columns of a data source with user-defined attributes and facts to enable the mapping of the business view, terms, and needs to the underlying database structure. You can have multiple MicroStrategy projects, each pointing to a data source. A database instance represents the connection to a data source, and one database instance could be referenced by multiple projects in a configuration. Each project contains a project schema, a set of tables and relationships that defined the logical model for that project.

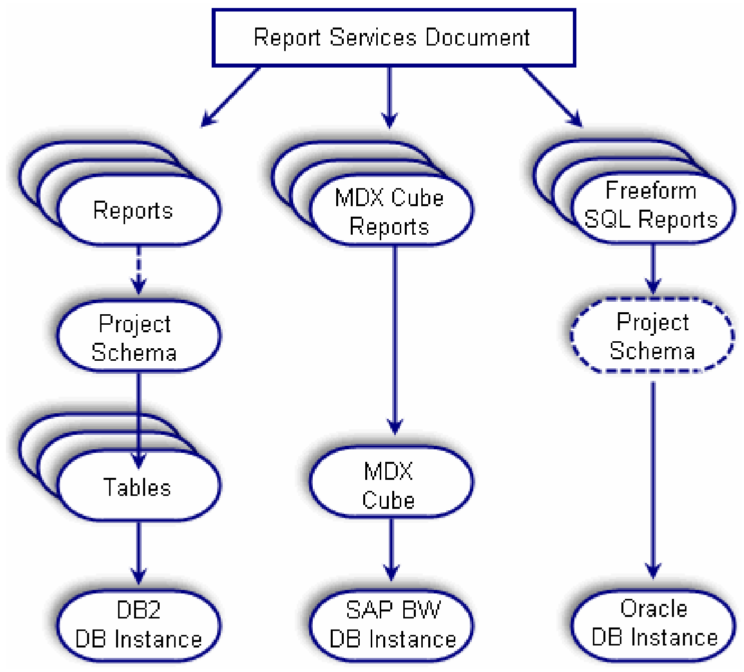
To learn more about how MDX cube sources can be used in addition to a project schema, see *The MicroStrategy Object Model*.

The MicroStrategy Object Model

The MicroStrategy model shows how a project can be extended to access MDX cube sources through a separate database instance. However, instead of pointing to the project schema, each MDX cube report points directly to one MDX cube in MicroStrategy, which is a logical placeholder for a physical cube that exists in an MDX cube source. Each report can only reference one specific MDX cube, due to the structure in MDX cube sources where queries can only be run against one physical cube at a time. You can create multiple reports to run against one MDX cube, and a single MicroStrategy project can reference multiple database instances, each of which can represent a distinct MDX cube source. For more information on how MicroStrategy can connect to MDX cube database instances along with database instances connected to your relational data warehouse, see *Configuring MDX Cube Database Instances in Projects, page 66*.

The MicroStrategy object model also shows how you can include any number of standard reports, Freeform SQL reports, Query Builder reports, and MDX cube reports in one Report Services document. By bringing these different types of reports together inside a document, report designers can create rich reports and analytics that take advantage of data from both data warehouses and MDX cube sources. You must import and map the data from MDX cubes into MicroStrategy in a manner that fits your data type conventions to support creating relationships between various MDX cube reports as well as between MDX cube reports and standard MicroStrategy reports (see *About Defining Column Data Types for MDX Cube Data, page 112*).

For information on Report Services documents, refer to the Document Creation Help and the Basic Reporting Help. For information on Freeform SQL and Query Builder reports, refer to the Advanced Reporting Help.



Relating Objects from MDX Cube Sources to MicroStrategy

This section describes how objects from the various MDX cube sources relate to and are converted into MicroStrategy objects. It is important to understand these relationship before you connect to your MDX cube source and import MDX cubes so that you are fully aware of how your MDX cube source data will be represented in MicroStrategy.

The following MDX cube sources and topics are covered:

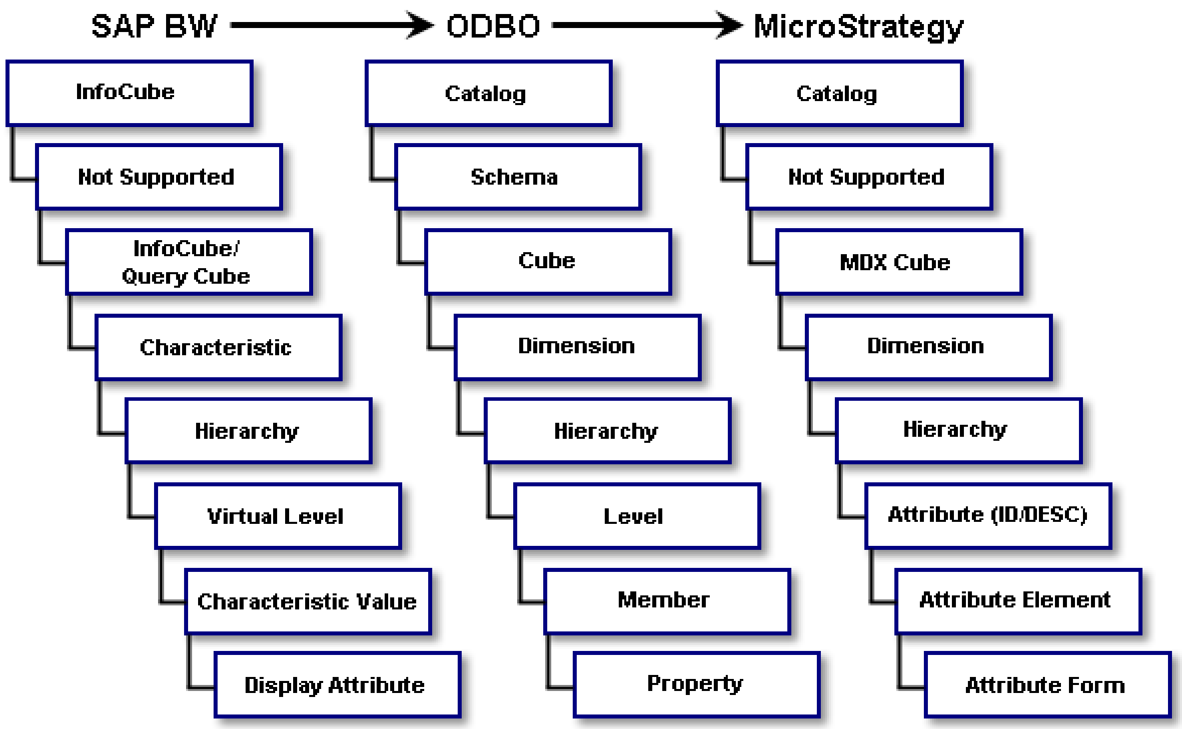
Relating Objects from SAP BW to MicroStrategy

MDX cube reports based off SAP BW MDX cubes can be treated like standard MicroStrategy reports.

The metadata translation process involves the following steps:

1. **From SAP BW to ODBO**: SAP exposes its query cubes and InfoCubes to Intelligence Server through the ODBO model. ODBO (OLE database for OLAP) is a protocol defined by Microsoft. ODBO defines an object model that is used in conjunction with MDX to query cubes. The ODBO model is similar to SAP's standard model, but not identical.
2. **From ODBO to MicroStrategy**: After SAP objects are translated into the ODBO model, they are translated into the MicroStrategy metadata model. You can then interact with SAP content while working within the paradigm that is consistent with the rest of MicroStrategy's products.

This diagram shows how SAP BW objects are exposed in ODBO and then how they are related to objects in the MicroStrategy environment.



Additional information and the process for each SAP BW object is described in the following topics:

SAP BW Terminology

The following list provides information on SAP BW objects for reference when *Relating Objects from SAP BW to MicroStrategy, page 10*:

For a comprehensive explanation of SAP BW objects, refer to your SAP documentation.

- **InfoObjects**: are the building blocks for individual cubes. They include objects such as characteristics and key figures, which are roughly equivalent to attributes and facts in a MicroStrategy project.
- **InfoProviders**: are all SAP BW data structures available for reporting and analysis purposes, such as the following:
  - **InfoCubes**: are multi-dimensional cubes are the primary objects that SAP BW uses to store data for analysis. InfoCubes define a specific domain of analysis in special areas, for example, finance or sales. Data is organized by dimension and stored physically in a star schema. The fact table at the center of an InfoCube contains the data available for analysis.
  - **ODS objects**: are operational data store objects. ODS objects are flat relational tables and are similar to MicroStrategy fact tables.
  - **MultiProviders**: are logical unions of two or more InfoProviders that are used to combine data from two different subject areas, for example, three InfoCubes or two ODS objects.
- **Query cubes (or query)**: defines a subset of data from an InfoCube or another InfoProvider. A query cube includes characteristics (dimensions/attributes) and key figures (metrics) from its source provider. The relationship between the query cube and the InfoCube is similar to the relationship between a MicroStrategy report and your data warehouse, in that a MicroStrategy report includes a subset of modeled attributes and metrics that are available in the data warehouse.

Query cubes generally offer better performance than InfoCubes because they are smaller and can be scheduled and cached within SAP BW. Query cubes also provide MicroStrategy users access to additional InfoProviders including ODS objects, InfoSets, and MultiProviders.

Any existing SAP BW query can be released for analysis within MicroStrategy. To release a query for analysis in MicroStrategy, choose the **Allow External Access to This Query** check box under the Extended tab in the SAP Query Properties dialog in the Query Analyzer interface. With this option enabled, report designers can quickly access existing query cubes and business content when working in MicroStrategy.

- **Characteristics**: provide classification possibilities for a dataset, such as sales region, product, customer group, fiscal year, and period. For example, a Sales Region characteristic can have North, Central, and South specifications.

  SAP BW characteristics are similar to MicroStrategy attributes. However, when each characteristic is translated into a cube, it is treated as a separate dimension for analysis. In addition, SAP BW hierarchies can be associated with a specific characteristic within SAP BW. These hierarchies are also available when you work with an MDX cube in MicroStrategy.

  SAP BW also has an object called an attribute, which is equivalent to an attribute form in MicroStrategy, not a MicroStrategy attribute.

- **Key figures**: describe numeric data, such as revenue, profit, and number of call centers. Key figures include amounts, quantities, numbers, dates, and times, all of which can be used in InfoCubes, ODS objects, and master data attributes. You can also create calculated key figures and restricted key figures in the query definition in the Business Explorer, similar to creating derived metrics and conditional metrics in MicroStrategy.

- **Hierarchies**: are objects that define the relationships among elements within a characteristic. For example, the Item characteristic might have a hierarchy that includes Category, Subcategory, and Item. This is a different paradigm from MicroStrategy's model where each attribute defines its own level, even though they are presented with the traditional attribute-based parent-child relationships when viewed in MicroStrategy.

- **Variables**: are used as parameters of a query in SAP BW. Defined in the Query Designer, variables can be of such types as characteristic values, hierarchies, hierarchy nodes, texts, and formulas. When a query is executed, these variables include values supplied by the system or by the user.

  When an MDX cube is imported into a MicroStrategy project, all the variables in the MDX cube are represented as MicroStrategy prompts. When the MDX cube is used to create a MicroStrategy MDX cube report, the MDX cube report inherits all of the prompts. See *Relating Objects from SAP BW to MicroStrategy, page 10* for more information on converting variables to prompts.

MicroStrategy lists all available MDX cubes for reporting and analysis in the MicroStrategy MDX Cube Catalog, including all of the published query cubes, InfoCubes, and MultiProviders. For information on using the MDX Cube Catalog to import MDX cubes, see *Importing MDX Cubes, page 83*.

Translating InfoCubes to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
InfoCube	Catalog	(Catalog)

- **SAP BW**: InfoCube

  Each InfoCube that has queries associated with it is exposed as a catalog in ODBO. Query cubes are accessed through their respective InfoCube catalogs.

- **ODBO**: Catalog

  Catalogs are used to group cubes. Therefore, ODBO catalogs are exposed in a few editors when choosing and managing cubes.

- **MicroStrategy**: (Catalog):

  Each catalog includes one InfoCube and associated query cubes, if any. Catalogs in MicroStrategy are represented in a folder.

Translating ODBO Schemas to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
Not supported	Schema	Not supported

- **SAP BW**: Not supported
- **ODBO**: Schema

  A schema in ODBO provides a grouping mechanism not supported by SAP BW or MicroStrategy.

- **MicroStrategy**: Not supported

Translating InfoCubes/Query Cubes to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
InfoCube/Query cube	Cube	MDX cube

- **SAP BW**: InfoCube/Query cube
- **ODBO**: Cube
- **MicroStrategy**: MDX cube

A MicroStrategy MDX cube is an object that is used to map the levels of an SAP BW cube into the MicroStrategy environment. MDX cubes can be thought of as similar to tables in the MicroStrategy metadata. In the same way that a metadata table maps the physical columns of a relational table to attributes and metrics, a MicroStrategy MDX cube maps the physical columns of an SAP BW cube to attributes and metrics. In MicroStrategy, attributes define the business context of data and metrics supply the data and calculations which can be combined on reports and documents to present your data. The MDX cube can be used to represent InfoCubes, MultiProviders, and query cubes.

Translating Characteristics to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
Characteristic	Dimension	Dimension

- **SAP BW**: Characteristic

  Characteristics in SAP BW are similar to attributes in MicroStrategy. For example, an InfoCube might include the Month characteristic, which represents months just like it does in a MicroStrategy attribute called Month.

  A characteristic appears as a dimension for MicroStrategy users. Each characteristic (dimension) has at least one hierarchy with two levels: the first level is an aggregate of all the data related to the characteristic, and the second level is the detailed data.

  When you import an MDX cube into MicroStrategy, you can control whether a characteristic's first level of aggregate data is imported and whether the second level is imported with a Level 01 suffix. For more information on setting import options for characteristics, see *Importing Levels and Suffixes for Characteristics, page 88*.

A characteristic can have any number of additional hierarchies, each with an arbitrary number of levels. The SAP BW characteristic hierarchies appear as MicroStrategy hierarchies to MicroStrategy users.

For example, in SAP BW you can build a Time hierarchy that is attached to the Month characteristic. The Time hierarchy defines a number of levels including Year, Quarter, and Month. These same levels could either be specifically defined as part of the hierarchy, or they could be other characteristics that are used to define the levels of this one hierarchy.

Shared dimensions allow a designer to use only one definition for a dimension across multiple cubes. Each characteristic in SAP is modeled as a dimension in ODBO and is shared across cubes. Therefore, all dimensions in cubes coming from SAP BW are shared.

- **ODBO**: Dimension

A dimension in ODBO defines a logical category of analysis. For example, Time and Geography are dimensions along which you can slice data.

Measures (metrics) are stored in a special measure dimension. In this way, measures are yet another dimension of a cube.

Measures in ODBO are called key figures in SAP BW, which are similar to metrics in MicroStrategy, and they are represented as physical columns.

If your SAP BW MDX cube source includes variables, there are times when these variables specify a name other than `[Measures]` for the measure dimension name. You can specify the measure dimension name in your SAP BW MDX cube source using the VLDB properties Name of Measure Dimension and MDX Remember Measure Dimension Name. For information on how to use these VLDB properties to specify the measure dimension name, refer to the *Supplemental Reference for System Administration*.

- **MicroStrategy**: Dimension

  A dimension object in MicroStrategy is similar to an ODBO dimension. It is used to group attributes and relate them to each other by defining parent-child relationships.

Translating Hierarchy Objects to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
Hierarchy	Hierarchy	Hierarchy

- **SAP BW**: Hierarchy
- **ODBO**: Hierarchy
- **MicroStrategy**: Hierarchy

  A hierarchy is used to group attributes (levels) together and define the relationships between these attributes. MicroStrategy uses hierarchy objects to represent both dimensions and hierarchies from ODBO.

Translating Virtual Levels to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
Virtual level	Level	Attribute

- **SAP BW**: Virtual level

  Levels are generated automatically based on either the definition of the characteristic or the hierarchies associated with a characteristic.

  SAP BW levels have names such as Region Level 01, Region Level 02, and so on. The inclusion of the term "Level" is an SAP BW convention. In MicroStrategy, architects have the option to rename the levels of a cube with a more user-friendly convention.

- **ODBO**: Level
- **MicroStrategy**: Attribute (ID/DESC)

  MicroStrategy attributes map to ODBO levels. Each ODBO level generates two physical columns and forms in MicroStrategy: ID and DESC.

Translating Characteristic Values to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
Characteristic value	Member	Attribute element

- **SAP BW**: Characteristic value
- **ODBO**: Member
- **MicroStrategy**: Attribute element

  Attribute elements are unique sets of attribute information which can come from either a database or an MDX cube imported from an MDX cube source. For example, 2020 and 2021 are attribute elements of the Year attribute.

Translating Characteristic Attributes to MicroStrategy

SAP BW --->	ODBO --->	MicroStrategy
Characteristic attribute	Property	Attribute form

- **SAP BW**: Characteristic attribute

  In SAP BW, forms are sometimes referred to directly as attributes. SAP BW also supports navigational attributes. These attributes are presented as distinct dimensions when working in MicroStrategy.

- **ODBO**: Property
- **MicroStrategy**: Attribute form

  Attribute forms provide additional information about a given attribute. For example, the Customer attribute may have the forms First Name and Last Name. This concept also applies to ODBO and SAP BW.

Converting SAP BW Variables into MicroStrategy Prompts

Variables in SAP BW allow users to enter values as parameters for the queries on a cube. Types of variables include characteristic values, hierarchies, hierarchy nodes, texts, and formula elements.

Variable types with the Customer Exit/SAP Exit and Authorization processing types are automatically resolved by the SAP BW system. Only variables with the Manual Entry/Default processing type are presented to users for resolution.

SAP BW variables of the type Replacement Path cannot be imported into MicroStrategy. If your SAP BW cube includes variables of the type Replacement Path, you must remove them before importing the cube into

MicroStrategy. Otherwise, an error occurs when you attempt to import the SAP BW cube.

Originally created in an SAP query cube, variables are represented as prompts in the MicroStrategy environment. MicroStrategy users answer prompts to complete the definition of a report at report execution time. The conversion process involves the following general steps:

1. When an SAP query cube is imported into a MicroStrategy project, variables are automatically turned into prompts in the MicroStrategy MDX cube.
2. When a MicroStrategy report is created using a MicroStrategy MDX cube, the report inherits the prompts included in the MDX cube.

In addition to the "inherited" variable prompts, standard MicroStrategy prompts can also be created for an MDX cube report. For more information, see *Prompts on MDX Cube Reports, page 165*.

The following table contains information on how the different types of SAP BW variables are mapped to MicroStrategy prompts.

SAP Variable Type	MicroStrategy Prompt	Notes
Characteristic Value variable	Element list prompt or attribute qualification prompt	Characteristic value variables offer an "Including/Excluding" option. Qualifications in the Including section cause the data to be included in the query, while those in the Excluding section restrict the data from being displayed in the query. To be consistent with the SAP functionality, the MicroStrategy interface qualifies on the key value of each element by default.

SAP Variable Type	MicroStrategy Prompt	Notes
Hierarchy variable	N/A	Not supported.
Hierarchy node variable	Hierarchy element list prompt	Both single and multiple selections are supported.
Text variable	Text value prompt	SAP text variables that provide dynamic names to SAP measures are converted to text value prompts in MicroStrategy. These prompts are then used to provide a dynamic metric alias for the MicroStrategy metric mapped to the SAP measure. When these MicroStrategy metrics are displayed on reports, it's metric name reflects the text value provided from the SAP text variable.
Formula variable	All types of value prompts	No major differences.

If you use any SAP BW key date variables in your query, you need to manually define the variable as a key date variable. See *Mapping SAP BW Variables to MicroStrategy Prompts, page 126* for information about how to define key date variables using the MicroStrategy MDX Cube Catalog while mapping your MDX cubes.

If you use any SAP BW variables with complex expression qualifications in your query, you can define which form is used to evaluate the variable's qualification, as described in *Mapping SAP BW Variables to MicroStrategy Prompts, page 126*.

Supporting SAP BW Structures

Structures in an SAP BW query cube define the two axes of a query (rows and columns). The two types of SAP BW structures are:

- Key figure structures, which are mapped to unique metrics in the MicroStrategy environment.
- Characteristic structures, which are represented as a single, flat dimension with one level mapped to a unique attribute in the MicroStrategy environment. This representation is consistent with how characteristic variables are represented in SAP BW through the OLAP Business Application Programming Interface (BAPI).

  In a MicroStrategy MDX cube report, you cannot drill down into the elements of characteristic structures.

When characteristic structures are imported into MicroStrategy as attributes, the structure element orders are preserved. This maintains a consistent view of your data across your SAP BW MDX cube source and MicroStrategy. Once imported into MicroStrategy, you can sort reports based on the structure element order. For information on sorting reports on structure element orders, see *Sorting Structure Elements and Preserving Order, page 175*.

Characteristic structures imported as attributes in MicroStrategy can also affect the value formats of metrics on MDX cube reports. For information on how you can specify that the metric values in MicroStrategy MDX cube reports inherit value formats from an MDX cube source, see *Inheriting MDX Cube Source Formats for Metric Values, page 178*.

Relating Objects from Oracle Essbase to MicroStrategy

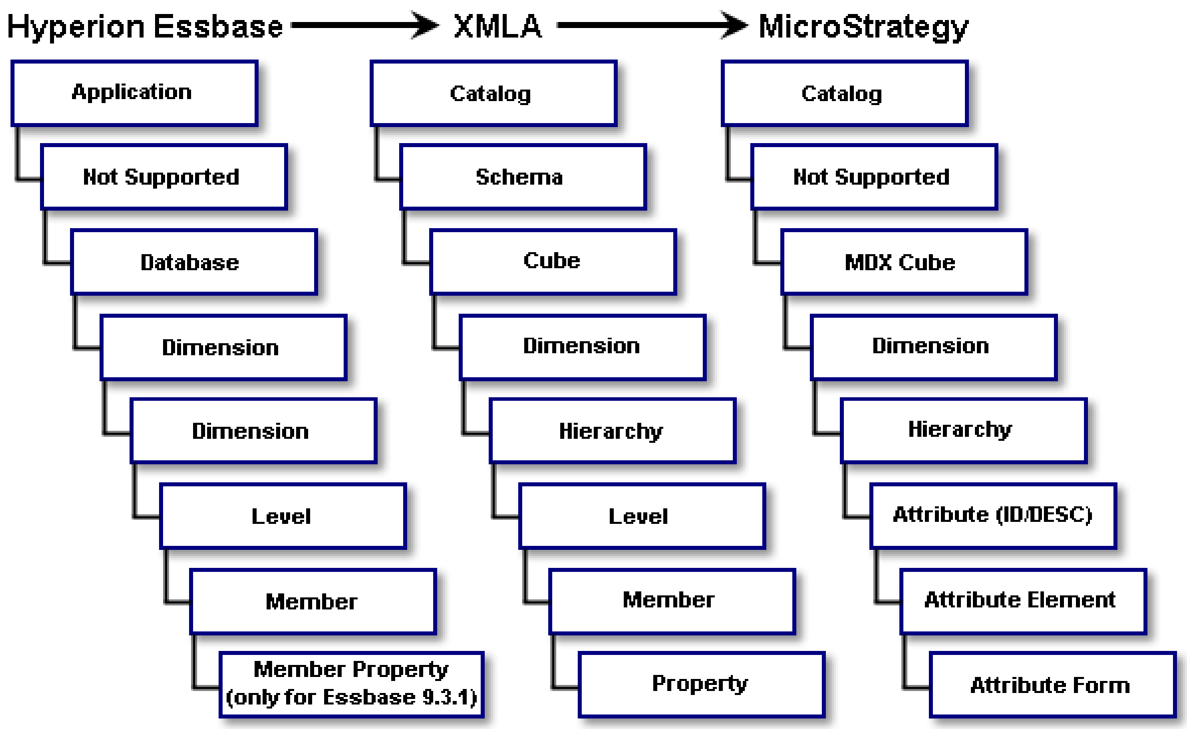
MDX cube reports from an Oracle Essbase 11.1.2 MDX cube can be treated like standard MicroStrategy reports.

In this section, the term Oracle Essbase is used to refer to Oracle Essbase 11.1.2.

The metadate model translation process involves the following steps:

1. **From Oracle Essbase to XMLA**: Oracle Essbase exposes its databases through the XMLA model which is derived from the ODBO model. XMLA defines an object model that is used in conjunction with MDX to query cubes. The Oracle Essbase model pre-dates XMLA so there are some differences. When thinking about Oracle Essbase objects, keep in mind how those objects appear in XMLA.
2. **From XMLA to MicroStrategy**: After Oracle Essbase objects are translated into the XMLA model, they are translated into the MicroStrategy metadata model. You can then interact with Oracle Essbase content while working within the paradigm that is consistent with the rest of MicroStrategy's products.

This diagram shows how Oracle Essbase objects are exposed in XMLA and then how they are related to objects in the MicroStrategy environment.



The process for each Oracle Essbase object is described in the following topics:

Translating Applications to MicroStrategy

Oracle Essbase --->	XMLA --->	MicroStrategy
Application	Catalog	(Catalog)

- **Oracle Essbase**: Application

  Each application is exposed as a catalog in XMLA. Cubes are accessed through their respective catalogs.

- **XMLA**: Catalog

  Catalogs are used to group cubes. Therefore, XMLA catalogs are exposed in editors when choosing and managing cubes.

- **MicroStrategy**: Catalog

  Each catalog includes one application and associated databases, if any. Catalogs in MicroStrategy are represented as a folder.

Translating XMLA Schemas to MicroStrategy

Oracle Essbase --->	XMLA --->	MicroStrategy
Not supported	Schema	Not supported

- **Oracle Essbase**: Not supported
- **XMLA**: Schema

  A schema in XMLA provides a grouping mechanism not supported by Oracle Essbase or MicroStrategy.

- **MicroStrategy**: Not supported

Translating Databases to MicroStrategy

Oracle Essbase --->	XMLA --->	MicroStrategy
Database	Cube	MDX cube

- **Oracle Essbase**: Database
- **XMLA**: Cube
- **MicroStrategy**: MDX cube

  A MicroStrategy MDX cube is an object that is used to map the levels of a Oracle Essbase cube into the MicroStrategy environment. MDX cubes are treated similarly to tables in the MicroStrategy metadata. A MicroStrategy MDX cube maps the physical columns of a Oracle Essbase cube to attributes and metrics in the same way that a metadata table maps the physical columns of a relational table to attributes and metrics. The MDX cube represents a Oracle Essbase database.

Translating Dimensions to MicroStrategy Dimensions

Oracle Essbase --->	XMLA --->	MicroStrategy
Dimension	Dimension	Dimension

- **Oracle Essbase**: Dimension

  In Oracle Essbase, a dimension represents the highest consolidation level in the database outline. The dimension is therefore both the highest level member in the dimension and the dimension itself. Each dimension has a single root node or member and is a child of the outline root node which is the database.

A Oracle Essbase dimension appears as a MicroStrategy dimension for MicroStrategy users. Each dimension has a single hierarchy with the number of levels determined by the greatest depth in the outline.

- **XMLA**: Dimension

  A dimension in XMLA defines a logical category of analysis. For example, Time and Geography are dimensions along which you can slice data.

  Measures (metrics) are stored in a special measure dimension. In this way, measures are yet another dimension of a cube.

  Measures in XMLA are the members of the dimension of type=Accounts in Oracle Essbase. These can be raw data or formulas with associated calculation or aggregation rules.

  When importing MDX cubes from Oracle Essbase, you can import the measures as a regular dimension. This can retain the structure of the measures as they exist in the Oracle Essbase MDX cube source. For information on how to import measures as a regular dimension, see *About Importing an Additional Measure Structure, page 90*.

  If the measures dimension in your Essbase MDX cube source uses a different name other than `[Measures]` you can specify the name in your Essbase MDX cube source using the VLDB properties Name of Measure Dimension and MDX Remember Measure Dimension Name. For information on how to use these VLDB properties to specify the measure dimension name, refer to the *Supplemental Reference for System Administration*.

- **MicroStrategy**: Dimension

  A dimension object in MicroStrategy is similar to an XMLA dimension. It is used to group attributes and relate them to each other by defining parent-child relationships.

Translating Dimensions to MicroStrategy Hierarchies

Oracle Essbase --->	XMLA --->	MicroStrategy
Dimension	Hierarchy	Hierarchy

- **Oracle Essbase**: Dimension

  An Oracle Essbase dimension is defined as part of the database outline. The outline is a hierarchical structure of database members with a parent containing its children. As a result, the outline defines a single hierarchy. Therefore, the dimension is the same as a MicroStrategy hierarchy.

- **XMLA**: Hierarchy
- **MicroStrategy**: Hierarchy

  Hierarchies are used to group attributes (levels) together and define the relationships between these attributes. MicroStrategy uses hierarchy objects to represent both dimensions and hierarchies from XMLA.

Translating Levels to MicroStrategy

Oracle Essbase --->	XMLA --->	MicroStrategy
Level	Level	Attribute

- **Oracle Essbase**: Level

  Levels group together members in a Oracle Essbase database outline.

Oracle Essbase levels may have default names such as Time.Levels(0). In MicroStrategy, architects have the option to rename the levels of a cube with a more user-friendly convention.

- **XMLA**: Level
- **MicroStrategy**: Attribute (ID/DESC)

  MicroStrategy attributes map to XMLA levels. Each XMLA level also generates two physical columns and the forms ID and DESC in MicroStrategy.

Translating Members to MicroStrategy

Oracle Essbase --->	XMLA --->	MicroStrategy
Member	Member	Attribute element

- **Oracle Essbase**: Member
- **XMLA**: Member
- **MicroStrategy**: Attribute element

  Element values come from either the database or a cube. For example, 2020 and 2021 are elements of the Year attribute.

Translating Member Properties to MicroStrategy

Oracle Essbase 11.1.2 --->	XMLA --->	MicroStrategy
Member property	Property	Attribute form

- **Oracle Essbase 11.1.2**: Member property

  In Oracle Essbase 11.1.2, a member property is a user-defined property that acts as a descriptive piece of information associated with an outline member.

- **XMLA**: Property

- **MicroStrategy**: Attribute form

  Attribute forms provide additional information about a given attribute. For example, the Customer attribute may have the forms First Name and Last Name.

Relating Objects from Analysis Services to MicroStrategy

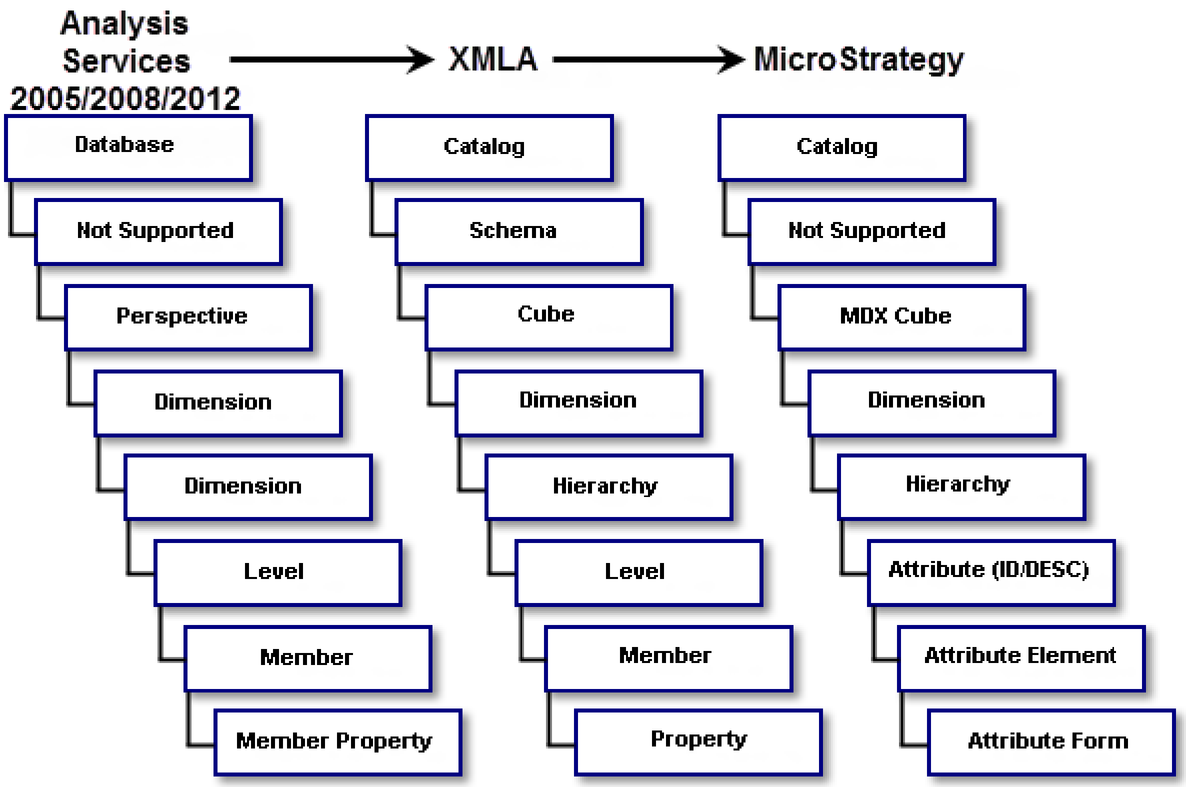
Microsoft Analysis Services 2012, 2014, 2016, 2017, 2019 and Azure (Analysis Services 2012/2014/2016/2017/2019/Azure) has a unique modeling approach for building cubes.

The translation process involves the following steps:

1. **From Analysis Services 2012/2014/2016/2017/2019/Azure to XMLA**: Analysis Services 2012/2014/2016/2017/2019/Azure exposes cubes through the XMLA model, which is derived from the ODBO model. XMLA defines an object model that is used in conjunction with MDX to query cubes.

2. **From XMLA to MicroStrategy**: After Analysis Services 2012/2014/2016/2017/2019/Azure objects are translated into the XMLA model, they are translated into the MicroStrategy metadata model. You can then interact with Analysis Services 2012/2014/2016/2017/2019/Azure content while working within the paradigm that is consistent with the rest of MicroStrategy's products.

This diagram shows how Analysis Services 2012/2014/2016/2017/2019/Azure objects are exposed in XMLA and then how they are related to objects in the MicroStrategy environment.



The process for each Analysis Services object is described in the following topics:

Translating Databases to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure -->	XMLA --->	MicroStrategy
database	Catalog	Catalog

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Database

  Each database is exposed as a catalog in XMLA. Cubes are accessed through their respective catalogs.

- **XMLA**: Catalog

Catalogs are used to group cubes. Therefore, XMLA catalogs are exposed in editors when choosing and managing cubes.

- **MicroStrategy**: Catalog

  Each catalog includes one database and associated cubes, if any. Catalogs in MicroStrategy are represented as a folder.

Translating XMLA Schemas to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure -->	XMLA --->	MicroStrategy
Not supported	Schema	Not supported

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Not supported
- **XMLA**: Schema

  A schema in XMLA provides a grouping mechanism not supported by Analysis Services 2012/2014/2016/2017/2019/Azure or MicroStrategy.

- **MicroStrategy**: Not supported

Translating Perspectives to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure -->	XMLA --->	MicroStrategy
Perspective	Cube	MDX cube

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Perspective

  A perspective in Analysis Services 2012/2014/2016/2017/2019/Azure is a view of the defined cube. The list of perspectives includes the original cube.

- **XMLA**: Cube
- **MicroStrategy**: MDX cube

  A MicroStrategy MDX cube is an object that is used to map the levels of an Analysis Services 2012/2014/2016/2017/2019/Azure cube into the MicroStrategy environment. MDX cubes are treated similarly to tables in the MicroStrategy metadata. In the same way that a metadata table maps the physical columns of a relational table to attributes and metrics, a MicroStrategy MDX cube maps the physical columns of an Analysis Services 2012/2014/2016/2017/2019/Azure cube to attributes and metrics. The MDX cube represents an Analysis Services 2012/2014/2016/2017/2019/Azure cube.

Translating Dimensions to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure --->	XMLA --->	MicroStrategy
Dimension	Dimension	Dimension

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Dimension

  In Analysis Services 2012/2014/2016/2017/2019/Azure, a dimension is defined from one or more data source tables. A data source does not always map directly to the tables in a relational database. All columns in the tables are eligible to become attributes of the dimension. Each attribute is used to define a hierarchy within the dimension and multi-level hierarchies can be defined as well.

An Analysis Services 2012/2014/2016/2017/2019/Azure dimension appears as a MicroStrategy dimension for MicroStrategy users. Each dimension represented in MicroStrategy can have one or more MicroStrategy hierarchies.

- **XMLA**: Dimension

  A dimension in XMLA defines a logical category of analysis. For example, Time and Geography are dimensions along which you can slice data.

  Measures (metrics) are stored in a special measure dimension. In this way, measures are yet another dimension of a cube.

  Measures in XMLA are the members of the Measures dimension in Analysis Services 2012/2014/2016/2017/2019/Azure. These can be columns in the data source table or calculated members represented by formulas with associated aggregation rules.

- **MicroStrategy**: Dimension

  A dimension object in MicroStrategy is similar to an XMLA dimension. It is used to group attributes and relate them to each other by defining parent-child relationships.

Translating Hierarchy Objects to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure --->	XMLA --->	MicroStrategy
Hierarchy	Hierarchy	Hierarchy

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Hierarchy

  Analysis Services 2012/2014/2016/2017/2019/Azure allows the definition of one or more hierarchies within a dimension as collections of attributes. The attributes become levels of the hierarchy.

- **XMLA**: Hierarchy
- **MicroStrategy**: Hierarchy

  Hierarchies are used to group attributes (levels) together and define the relationships between these attributes. MicroStrategy uses hierarchy objects to represent both dimensions and hierarchies from XMLA.

Translating Levels to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure --->	XMLA --->	MicroStrategy
Level	Level	Attribute

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Level

  Each attribute in a hierarchy becomes a level.

- **XMLA**: Level
- **MicroStrategy**: Attribute (ID/DESC)

  MicroStrategy attributes map to XMLA levels. Each XMLA level generates two physical columns and forms in MicroStrategy: ID and DESC.

Translating Members to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure --->	XMLA --->	MicroStrategy
Member	Member	Attribute element

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Member
- **XMLA**: Member
- **MicroStrategy**: Attribute element

  Element values come from a cube. For example, 2020 and 2021 are elements of the Year attribute.

Translating Member Properties to MicroStrategy

Analysis Services 2012/2014/2016/2017/2019/Azure --->	XMLA --->	MicroStrategy
Member property	Property	Attribute form

- **Analysis Services 2012/2014/2016/2017/2019/Azure**: Member property

  Attributes can be related as member properties when defining the levels of a dimension. Member properties are returned in the XMLA property schema rowset.
- **XMLA**: Property
- **MicroStrategy**: Attribute form

Attribute forms provide additional information about a given attribute. For example, the Customer attribute may have the forms First Name and Last Name.

Relating Objects from TM1 to MicroStrategy

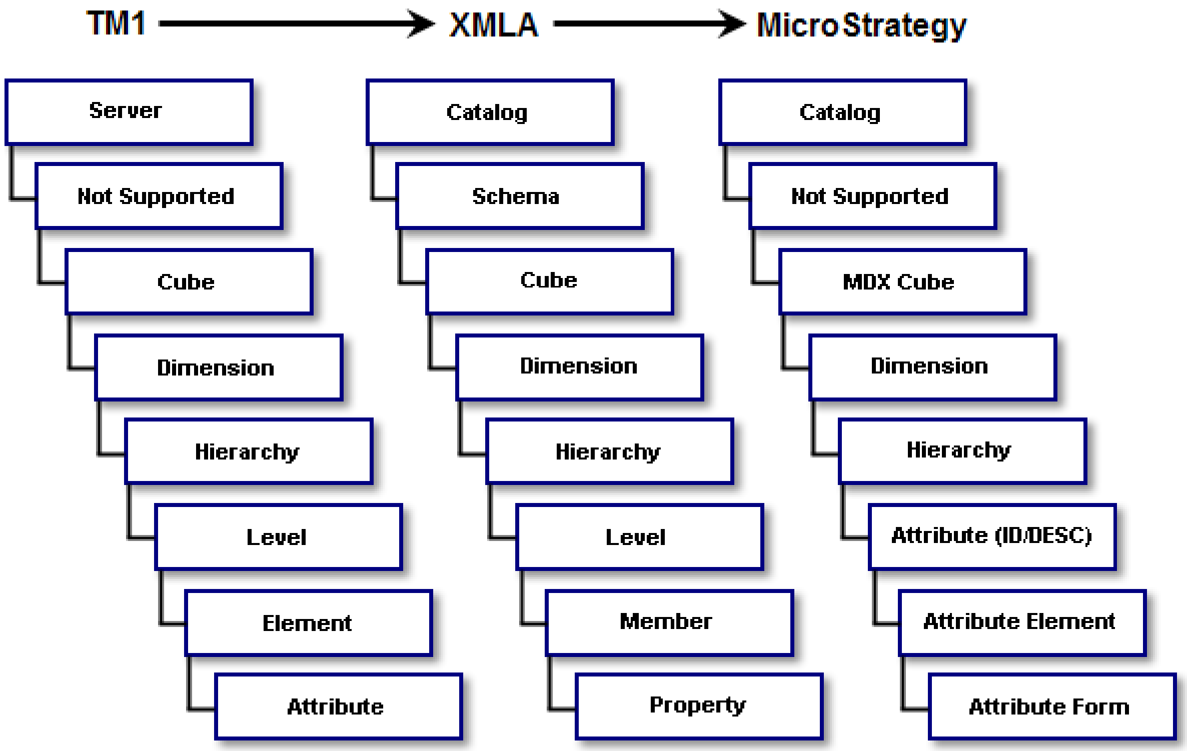
TM1 has a unique modeling approach for building cubes.

The translation process involves the following steps:

1. **From TM1 to XMLA**: TM1 exposes cubes through the XMLA model which is derived from the ODBO model. XMLA defines an object model that is used in conjunction with MDX to query cubes.

2. **From XMLA to MicroStrategy**: After TM1 objects are translated into the XMLA model, they are translated into the MicroStrategy metadata model. You can then interact with TM1 content while working within the paradigm that is consistent with the rest of MicroStrategy's products.

This diagram shows how TM1 objects are exposed in XMLA and then how they are related to objects in the MicroStrategy environment.



The process for each TM1 object is described in the following topics:

Translating Servers to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Server	Catalog	Catalog

- **TM1**: Server

  Each TM1 server is exposed as a catalog in XMLA. Cubes are accessed through their respective catalogs.

- **XMLA**: Catalog

  Catalogs are used to group cubes. Therefore, XMLA catalogs are exposed in editors when choosing and managing cubes.

- **MicroStrategy**: Catalog

  Each catalog includes one database and any associated cubes. Catalogs in MicroStrategy are represented as a folder.

Translating XMLA Schemas to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Not supported	Schema	Not supported

- **TM1**: Not supported

- **XMLA**: Schema

  A schema in XMLA provides a grouping mechanism not supported by TM1 or MicroStrategy.

- **MicroStrategy**: Not supported

Translating Cubes to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Cube	Cube	MDX cube

- **TM1**: Cube
- **XMLA**: Cube
- **MicroStrategy**: MDX cube

  A MicroStrategy MDX cube is an object that is used to map the levels of a TM1 cube into the MicroStrategy environment. MDX cubes are treated similarly to tables in the MicroStrategy metadata. In the same way that a metadata table maps the physical columns of a relational table to attributes and metrics, a MicroStrategy MDX cube maps the physical columns of a TM1 cube to attributes and metrics. The MDX cube represents a TM1 cube.

Translating Dimensions to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Dimension	Dimension	Dimension

- **TM1**: Dimension

  In TM1, a dimension is defined from one or more data source tables. All columns in the tables are eligible to become attributes of the dimension.

Each attribute is used to define a hierarchy within the dimension and multi-level hierarchies can be defined as well.

A TM1 dimension appears as a MicroStrategy dimension for MicroStrategy users. Each dimension represented in MicroStrategy can have one or more MicroStrategy hierarchies.

- **XMLA**: Dimension

  A dimension in XMLA defines a logical category of analysis. For example, Time and Geography are dimensions along which you can slice data.

  Measures (metrics) are stored in a special measure dimension. In this way, measures are yet another dimension of a cube.

  Measures in XMLA are the members of the Measures dimension in TM1. These can be columns in the data source table or calculated members represented by formulas with associated aggregation rules.

  If the measures dimension in your TM1 MDX cube source uses a different name other than `[Measures]` you can specify the name in your TM1 MDX cube source using the VLDB properties Name of Measure Dimension and MDX Remember Measure Dimension Name. For information on how to use these VLDB properties to specify the measure dimension name, refer to the *Supplemental Reference for System Administration*.

- **MicroStrategy**: Dimension

  A dimension object in MicroStrategy is similar to an XMLA dimension. It is used to group attributes and to relate them to each other by defining parent-child relationships.

Translating Hierarchy Objects to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Hierarchy	Hierarchy	Hierarchy

- **TM1**: Hierarchy

  TM1 allows the definition of one or more hierarchies within a dimension as collections of attributes. The attributes become levels of the hierarchy.

- **XMLA**: Hierarchy
- **MicroStrategy**: Hierarchy

  Hierarchies are used to group attributes (levels) together and define the relationships between these attributes. MicroStrategy uses hierarchy objects to represent both dimensions and hierarchies from XMLA.

Translating Levels to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Level	Level	Attribute

- **TM1**: Level

  Each attribute in a hierarchy becomes a level.

- **XMLA**: Level
- **MicroStrategy**: Attribute (ID/DESC)

  MicroStrategy attributes map to XMLA levels. Each XMLA level generates two physical columns and forms in MicroStrategy: ID and DESC.

Translating Elements to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Element	Member	Attribute element

- **TM1**: Element
- **XMLA**: Member
- **MicroStrategy**: Attribute element

  Element values come from a cube. For example, 2009 and 2010 are elements of the Year attribute.

Translating Attributes to MicroStrategy

TM1 --->	XMLA --->	MicroStrategy
Attribute	Property	Attribute form

- **TM1**: Attribute
- **XMLA**: Property
- **MicroStrategy**: Attribute form

  Attribute forms provide additional information about a given attribute. For example, the Customer attribute may have the forms First Name and Last Name.

Connecting to MDX Cube Sources

While MicroStrategy handles most of the conversion processes necessary to integrate with MDX cube source data, you must establish the connection between MicroStrategy and your MDX cube source. MicroStrategy connects to SAP BW using SAP's OLAP Business Application Programming Interface (BAPI) and to other sources using XML for Analysis (XMLA). These configurations require modifications to projects, project sources, database instances, and other objects in MicroStrategy.

The following sections explain how to connect to different sources and how to configure and authenticate each MDX cube database instance:

For troubleshooting and diagnostics related to MDX cube sources, see the *Troubleshooting* section of the System Administration Help.

Connecting to SAP BW servers

Before creating any reports using the SAP BW data, you need to establish a connection between MicroStrategy and the SAP BW system. MicroStrategy certifies connecting to SAP BW 7.3 and 7.4.

With the SAP BW Certification on MicroStrategy, MicroStrategy Intelligence Server is certified to connect to and execute reports against SAP BW cubes. As SAP's proprietary API for accessing SAP BW data and functionality, the OLAP BAPI provides an open interface through which Intelligence Server can access the SAP BW data.

MicroStrategy Web is certified to run on SAP Web Application Server, and MicroStrategy Web and SDK are certified to run with SAP Enterprise Portal through iView Packages.

See the following topics for prerequisites and detailed connection steps:

Prerequisites for Connecting to SAP BW Servers

Before installing on either Windows or Linux, ensure that you have the following access and permissions:

- A valid login for the SAP Service Marketplace to download the SAP Java Connector files.

  If you are using MicroStrategy Intelligence Server on Linux, you also need Write privileges to the `/install/jar` installation directory on your Linux machine.

- MicroStrategy login with administrative privileges and access to a Windows machine with MicroStrategy Developer, both of which are necessary to create a database instance.

- A SAP BW user login which will be used to connect to your SAP BW server, with remove function call (RFC) authorization to the groups `RFC1`, `RSOB`, `SDIFRUNTIME`, and `SYST`.

- The following query and InfoCube access is required:

  For each query to allow access to, you must have `EXECUTE` access for the `S_RS_COMP` authorization object.

  For query object types, you must have `EXECUTE` access for the `S_RS_ICUBE` authorization object.

  For each InfoCube to allow access to, you must have activity `DISPLAY` access and InfoCube subobject `DATA` access for the `S_RS_ICUBE` authorization object.

Download and Install the Java Connector

After reviewing the *Prerequisites for Connecting to SAP BW Servers*, begin the connection process by downloading and installing the Java Connector on either Windows or Linux:

If you install the Java Connector on Linux, you will still need to complete the remaining steps to connect to SAP BW servers using MicroStrategy Developer on Windows.

Download and Install the Java Connector on Windows

After reviewing the *Prerequisites for Connecting to SAP BW Servers*, begin the connection process as follows:

1. Open the SAP Service Marketplace and download the SAP Java Connector.

   MicroStrategy supports the use of version 3.x. You must download and install the 64-bit version of the 3.x Java Connector for Intelligence Servers hosted on a 64-bit machine.

   As part of the SAP Java Connector installation, you are required to install the new Visual Studio .NET C/C++ run-time libraries on Windows platforms. See SAP Note 684106 for details on how to install them.

   For information on what SAP Java Connector you need to support your SAP BW system, refer to your third-party SAP BW documentation.

2. Install the SAP Java Connector.

3. Place the following `Sapjco3.jar` and `Sapjco3.dll` files in the MicroStrategy Common Files folder.

   The default location of this folder is `C:\Program files\Common files\MicroStrategy`. This folder must be referenced in the system's `Path` environment variable.

   To verify that the directory is included in the value for the `Path` variable, view the `Path` environment by right-clicking on **My Computer** and choosing **Properties**> **Advanced** and choose **Environment**

Variables. In the list of **System Variables**, locate `Path`. The directory should be listed in the environment variable.

Upon completing the installation, see *Create a Database Instance for Your SAP BW Connection* for next steps.

Download and Install the Java Connector on Linux

After reviewing the *Prerequisites for Connecting to SAP BW Servers*, begin the connection process as follows:

1. Go to the SAP Java Connector page of the SAP Support Portal and download the SAP Java Connector.

   MicroStrategy supports the use of version 3.x. You must download and install the 64-bit version of the 3.x Java Connector for Intelligence Servers hosted on a 64-bit machine.

   For information on what SAP Java Connector you need to support your SAP BW system, refer to your third-party SAP BW documentation.

2. Choose the zip file compatible with your environment and unzip it. For example, use the command `gunzip[`*`file name`*`]` or `gzip [`*`file name`*`]`.

3. Search for the files listed in the following table, copy them onto your machine, and create a new directory for them. For example, `/var/opt/MicroStrategy/SAP`.

SUN	Linux
`libsapjco3.so`	`libsapjco3.so`
`sapjco3.jar`	`sapjco3.jar`

4. Add `sapjco.jar` or `sapjco3.jar` to the MicroStrategy installation directory `[HOME_PATH]/install/jar`.

5. Add `libsapjco3.so` to the MicroStrategy installation directory `[HOME_PATH]/install/lib`.Edit the `SAP.sh` file in the MicroStrategy installation directory `[INSTALL_PATH]/env/SAP.sh` by performing the following steps:

   a. Add the Write and Execute privileges to this file. The default is Read Only. You can type the command "`chmod +wx SAP.sh`" in the Linux console.

   b. Open the `SAP.sh` file and enter the information for `MSTR_SAP_LIBRARY_PATH=''`. This information indicates where the server needs to point to use the downloaded files.

      In the example below, you must replace *`DIRECTORY_PATH`* with the path to the directory where you installed the SAP Java Connector files. The default directory path is `/var/opt/MicroStrategy/SAP`.

      For example, for **Linux**:

      ```
      #
      # set up the environment for SAP
      #
      MSTR_SAP_LIBRARY_PATH='DIRECTORY_PATH'

      if [ "${MSTR_SAP_LIBRARY_PATH}" != 'DIRECTORY_
      PATH' ]; then
      mstr_append_path LD_LIBRARY_PATH
      "${MSTR_SAP_LIBRARY_PATH:?}"
      export LD_LIBRARY_PATH
      fi
      ```

   c. Save the file.

Create a Database Instance for Your SAP BW Connection

After you completed the steps described in *Download and Install the Java Connector on Windows* or *Download and Install the Java Connector on*

Linux, create a database instance as follows:

1. In Developer, navigate to the Folder List.
2. Open your project source and expand **Administration**.
3. Choose **Configuration Managers** > **Database Instance**.
4. Click **File**> **New**> **Database Instance** to open the Database Instance Editor.
5. In the **Database instance name** text field, type a descriptive name for the database instance.
6. Choose **Database connection type** > **SAP BW 7.3** or **SAP BW 7.4**, depending on your version of SAP BW.

   If you connect to SAP BW 7.01 SP02, you can create MDX cube reports that include more than one million cells. MicroStrategy also recommends using the 3.x version of the SAP Java Connector to fully support this scalability.

7. From the **Advanced** tab, choose the version of the SAP Java Connector you downloaded at the beginning of these steps. MicroStrategy supports the use of the 3.x version of the SAP Java Connector.

   The 2.x version of the SAP Java Connector, was supported in pre-10 versions of MicroStrategy. You can update your database instances that previously used version 2.x to use version 3.x.

After creating the database instance, see *Create an SAP BW Database Connection* for next steps.

Create an SAP BW Database Connection

After you completed the steps described in *Create a Database Instance for Your SAP BW Connection*, create a database connection:

1. Choose **General** > **New**.
2. Provide the following information from your SAP Logon:

- **Database connection name**: This is the name to distinguish the database connection from database connections for other database instances.
- **Logon method**: Determines the type of SAP BW system you are connecting to. You have the following options:

  Application server: Connect to a specific SAP BW application server. You must provide the following information to connect to an SAP BW application server:

  - **Application Server**: Name or IP address of the SAP BW Server.
  - **SAP Router String**: SAP router string, which is only required if you use an SAP Router.
  - **System Number**: System number from the SAP BW system.
  - **Client Number**: Client number from the SAP BW system.
  - **Language**: Language code provided by your SAP administrator. For example, EN is the language code for English.

  Message server: Connect to an SAP BW message server, which acts as a load balancing mechanism for a cluster of SAP BW application servers. Once connected to the SAP BW message server, the SAP BW system can its load balancing capabilities to connect to an SAP BW application server. You must provide the following information to connect to an SAP BW message server:

  - **Message Server**: Name or IP address of the SAP BW Server.
  - **SAP Router String**: SAP router string, which is only required if you use an SAP Router.
  - **System ID**: System ID from the SAP BW system.

- **Logon group**: SAP BW logon group that determines what SAP BW application servers in the cluster can be accessed.
- **Client Number**: Client number from the SAP BW system.
- **Language**: Language code provided by your SAP administrator. For example, EN is the language code for English.

Use SNC Authentication: Secure Network Communication (SNC) integrates SAPNetWeaver Single Sign-On or an external security product. SNC protects the data communication paths between the various client and server components of the SAP system that use the SAP protocols RFC or DIAG.

- **SNC Partner Name**: Secure Network Communication name of the communication partner (SAP NetWeaver AS ABAP).
- **Level of Protection**: Select one from the following quality of protections:
  - **Authentication only**: The system verifies the identity of the communication partners. This is the minimum protection level offered by SNC.
  - **Integrity protection**: The system detects any changes or manipulation of the data, which may have occurred between the two end points of a communication.
  - **Privacy protection**: The system encrypts the messages being transferred to make eavesdropping useless. Privacy protection also includes integrity protection of the data. This is the maximum level of protection provided by SNC.
- **SNC My Name**: Secure Network Communication name of the adapter.
- **SNC Library Path**: Path of the library file, `sacpcrypto.dll`.

- **SNC Login**: Use either the project user login or the database connection user login.

  If you choose the warehouse login, the value you enter in the following User field is used to access the warehouse. The User used to log in to the MicroStrategy project is not used.

  If User login is selected, the MicroStrategy project login is used.

  You can also directly edit the connection string to enable SNC authentication with parameters. For example:

  ```
  APPSERVER=10.27.10.200;SYSNR=12;CLIENT=100;LANG=EN;JCO
  ;SNC;SNC_LIB=C:\SAP\sapcrypto.dll;SNC_
  MYNAME=p:CN=MSTRService NSP, CN=Users, DC=CORP,
  DC=microstrategy, DC=com;SNC_
  PARTNERNAME=p:CN=mb2adm@LABS.MICROSTRATEGY.COM;SNC_
  LOGINUSER=0;SNC_QOP=1;SNC_MODE=1;
  ```

  The Data Source dialog is not available in MicroStrategy Developer. You can however edit the connection string to enable SNC authentication in Developer.

After creating the database connection, see *Complete the Connection* for next steps.

Complete the Connection

After you completed the steps described in *Create an SAP BW Database Connection*, create a database connection as follows:

MicroStrategy requires that the SAP BW user login used to connect to your SAP BW server has remote function call (RFC) authorization to the `SDIFRUNTIME` group. If necessary, contact your SAP administrator to grant your SAP BW user login RFC authorization to the `SDIFRUNTIME` group.

You can configure your SAP BW environment to allow users single-sign on access to MicroStrategy through the use of integrated authentication. If you configure this type of authentication, as described in the System Administration Help, the database login is required but is not used for authentication.

1. Click **New** to create a database login with the user and password credentials used to connect to SAP BW.
2. Enter a name for the database login object, login ID, and password.
3. Click **OK**.
4. Choose **Default database login name** > **OK**.
5. Choose **Database connection** > **OK**.

After you have connected a database instance to your SAP BW server, you can begin accessing your SAP BW data from a MicroStrategy project. For more information, see *Configuring MDX Cube Database Instances in Projects, page 66*.

Connecting to Oracle Essbase Servers

Before creating any reports using the Oracle Essbase data, you need to establish a connection in MicroStrategy to the Oracle Essbase 11.1.2 servers.

To begin the process of connecting MicroStrategy to your Oracle Essbase servers, complete the following installations and configurations:

Install the MDX Cube Provider and the Essbase Client

Begin the Oracle Essbase connection process with the following installations:

1. Install the MicroStrategy MDX Cube Provider on a machine. For information on how to install this component, which is provided with MicroStrategy, see the Installation and Configuration Help.

2. Install the Essbase client, including the Essbase C API, on the machine where you installed MicroStrategy and the MicroStrategy MDX Cube Provider. For 32-bit operating systems, install the 32-bit Essbase client, and for 64-bit operating systems, install the 64-bit Essbase client.

   For information on how to install the Essbase client libraries, refer to your third-party Oracle documentation.

   > Using Essbase version 11.1.2.2 can cause connection issues. To avoid this, upgrade Essbase Analytical Provider Services to version 11.1.2.4 to support integration with the MicroStrategy MDX Cube Provider.

Upon completing the installations, see *Configure the Essbase Client and the MDX Cube Provider* for next steps.

Configure the Essbase Client and the MDX Cube Provider

After you complete the installations described in *Install the MDX Cube Provider and the Essbase Client*, configure them as follows:

1. After installing the Essbase client and Essbase C API, you must create various environment variables. MicroStrategy provides scripts to create the required environment variables.

   Using a command line interface, navigate to the MicroStrategy Common Files folder and execute one of the following scripts:

   - For 64-bit Essbase clients:

     ```
     EssbaseConnector_SetEnv_64_64.cmd
     ```

   - For 32-bit Essbase clients:

     ```
     EssbaseConnector_SetEnv_32_32.cmd
     ```

2. In the installation location for the MicroStrategy MDX Cube Provider, you must modify the `Datasources.xml` file. We recommend making a backup copy before modifying this .xml file.

a. In the `Datasources.xml` file, locate the `<AuthenticationMode></AuthenticationMode>` section.

   Within this section, type the value `EssbaseBasic`.

b. In the `Datasources.xml` file, locate the `<ProgramID></ProgramID>` section.

   Within this section, type the value `ESSBASE`.

   The `Datasources.xml` file includes an example definition for the `<DataSources></DataSources>` section with default values for an Essbase connection. You can use this section as a template for the `<DataSources>` definition of your Oracle Essbase connection.

Upon completing the configurations, see *Verify the MDX Cube Provider and Oracle Essbase Configurations* for next steps.

Verify the MDX Cube Provider and Oracle Essbase Configurations

After you complete the steps described in *Configure the Essbase Client and the MDX Cube Provider*, verify the configurations as follows before creating the database instance for the connection.

1. Install Oracle Essbase Provider Services on the application server machine to verify that access is available to the Oracle Essbase server via the service console. For information on installation procedures, refer to the Essbase documentation.
2. By default, access to the application uses standard authentication. By setting up a MicroStrategy connection using your Oracle Essbase login and password, you can use your Oracle Essbase login and password through Provider Services.

3. Verify the configuration is working by connecting to the provider URL from your browser. You should receive a confirmation from the provider that includes a display of currently configured properties.

   You can perform a test of the connection to your MDX cube servers, separate from any MicroStrategy dependencies, using the XMLA Connectivity Test Tool provided with your MicroStrategy installation. In Windows environments, this tool can be used by running the `XMLATest.exe` file included in the MicroStrategy common files. In UNIX environments, this tool can be used by running the command `mstrxmlatest` from the `/bin` directory within your MicroStrategy home path.

Upon completing these verifications, see *Create a Database Instance for Oracle Essbase 11.1.2* for next steps.

Create a Database Instance for Oracle Essbase 11.1.2

After you verify the configurations as described in *Verify the MDX Cube Provider and Oracle Essbase Configurations*, create a database instance as follows:

1. From the MicroStrategy Developer Folder List, connect to your project source.
2. Choose **Administration** > **Configuration Managers** > **Database Instance**.
3. Choose **File** > **New** > **Database Instance**.
4. In the **Database instance name** field, type a descriptive name for the database instance.
5. From the **Database connection type** drop-down list, choose **Oracle Essbase 11.1.2 MDX Cube Provider**.

Upon creating the database instance, see *Create Database Connection Parameters for Oracle Essbase* for next steps.

Create Database Connection Parameters for Oracle Essbase

After you *Create a Database Instance for Oracle Essbase 11.1.2*, create the database connection as follows:

1. Click **New**.
2. Provide the following information:

   Database connection name: This is the name to distinguish the database connection from database connections for other database instances.

   URL: This is the URL of the provider that was configured for HTTP access. For example: `http://`*`fully-qualified-machinename`*`/MicroStrategyMDX/MicroStrategyMDX.asmx`.

   > The *`fully-qualified-machinename`* is usually of the form `machine.domain.company.com`. You can also use the IP address as the *`fully-qualified-machinename`*. For Oracle Essbase the URL is usually case sensitive.

   DSI: The DataSourceInfo (DSI) value is *`<machine name>`*.

   Catalog: The Essbase Catalog value is the Oracle Essbase application containing the database to connect to with MicroStrategy. Use the Essbase Administration Console to view the applications and databases available on the server.

Once you have created the database connection, see *Complete the Connection to Oracle Essbase* for next steps.

Complete the Connection to Oracle Essbase

After completing the steps described in *Create Database Connection Parameters for Oracle Essbase*, complete the following steps to save your changes and complete the connection:

1. Click **New**.
2. Enter a name for the database login object, login ID, and password.

   The login ID and password must be a valid Oracle Essbase user and password.

3. Click **OK**.
4. Choose **Default database login name** > **OK**.
5. Choose **Database connection** > **OK**.

After you have connected a database instance to your Oracle Essbase server, you can begin accessing your Oracle Essbase data from a MicroStrategy project. For information on configuring the MDX cube database instance in a project, see *Configuring MDX Cube Database Instances in Projects, page 66*.

Connecting to Analysis Services Servers

Before creating any reports using the Analysis Services data, you need to establish a connection to the Analysis Services servers.

Creating a database instance and its components requires a MicroStrategy login with administrative privileges.

To begin the process of connecting MicroStrategy to your Analysis Services servers, complete the following installations and configurations:

Install and Configure Analysis Services and the MDX Cube Provider

Install and configure Analysis Services 2012/2014/2016/2017/2019/Azure as follows to meet the deployment and security settings required to enable the connection to MicroStrategy:

1. Install Microsoft Analysis Services 2012/2014/2016/2017/2019/Azure. Refer to the Microsoft documentation for more information.

   Information for correctly installing Analysis Services 2012/2014/2016/2017/2019/Azure and any other Microsoft products can be found in your Microsoft documentation.

2. Install the MicroStrategy MDX Cube Provider on a machine. For information on how to install this component, which is provided with MicroStrategy, see the Installation and Configuration Help.

3. Install the Microsoft Analysis Services OLE DB Provider on the machine where you installed MicroStrategy and the MicroStrategy MDX Cube Provider. The version that you install depends on your operating system's architecture:

   If you install on a 32-bit operating system, install the 32-bit Microsoft Analysis Services OLE DB Provider. MicroStrategy supports the Microsoft Analysis Services OLE DB Provider 2012 and newer versions.

   or

   If you install on a 64-bit operating system, install the 64-bit Microsoft Analysis Services OLE DB Provider.

   If you are connecting to Microsoft Analysis Services 2014, you must install the 2012 SP2 version of the Microsoft Analysis Services OLE DB Provider.

4. In the installation location for the MicroStrategy MDX Cube Provider, you must modify the `Datasources.xml` file. We recommend making a backup copy before modifying this .xml file.

   - In the `Datasources.xml` file, locate the `<AuthenticationMode></AuthenticationMode>` section. Within this section, type the authentication mode used for the virtual directory running on Microsoft Internet Information Services. You can use either Basic (`MSOLAPBasic`) or Integrated (`MSOLAPIntegrated`) authentication. For information on these authentication options, refer to your third-party Microsoft documentation.

- In the `Datasources.xml` file, locate the `<ProgramID></ProgramID>` section. Within this section, type the value `MSOLAP`.

  The `Datasources.xml` file includes an example definition for the `<DataSources></DataSources>` section with default values for a Microsoft Analysis Services connection. You can use this section as a template for the `<DataSources>` definition of your Microsoft Analysis Services connection.

5. Create one or more Microsoft Analysis Services role definitions along with mapped NT user logins. If anonymous access is enabled in the XMLA virtual directory, the Windows user associated with anonymous access must have membership to one of the security roles defined in Analysis Services.
6. You must either configure Kerberos authentication protocol for credential delegation or define an anonymous user on the IIS XMLA provider virtual directory. The Windows user associated with anonymous access must have membership in one of the security roles defined in Analysis Services.
7. In order to enable Kerberos authentication pass-through to Analysis Services, you must install the MicroStrategy MDX Cube provider on the same machine as the MicroStrategy Intelligence Server.

You can perform a test of the connection to your MDX cube servers, separate from any MicroStrategy dependencies, using the XMLA Connectivity Test Tool provided with your MicroStrategy installation. In Windows environments, this tool can be used by running the

`XMLATest.exe` file included in the MicroStrategy common files. In Linux environments, this tool can be used by running the command `mstrxmlatest` from the `/bin` directory within your MicroStrategy home path.

Upon completing the installations and configurations, see *Creating a Database Instance for Analysis Services 2012/2014/2016/2017/2019/Azure* for next steps.

Creating a Database Instance for Analysis Services 2012/2014/2016/2017/2019/Azure

After you complete the installation and configuration as described in *Install and Configure Analysis Services and the MDX Cube Provider*, create a database instance as follows:

1. In MicroStrategy Developer, navigate to the Folder list.
2. Open your project source and choose **Administration**.
3. Choose **Configuration Managers** > **Database Instance.**
4. Choose **File** > **New** > **Database Instance**.
5. In the **Database instance name** text field, type a descriptive name for the database instance.
6. From the **Database connection type** drop-down, choose:

   Microsoft Analysis Services 2016/2017/2019/Azure (MDX Cube Provider) if you are connecting to Analysis Services 2016/2017/2019/Azure with the MicroStrategy MDX Cube Provider.

   or

   Microsoft Analysis Services 2012 and 2014 (MDX Cube Provider) if you are connecting to Analysis Services 2012, Analysis Services 2012 in Tabular Mode, or Analysis Services 2014 with the MicroStrategy MDX Cube Provider.

Upon creating the database instance, see *Create a Database Connection for Analysis Services* for next steps.

Create a Database Connection for Analysis Services

After creating a database instance as described in *Creating a Database Instance for Analysis Services 2012/2014/2016/2017/2019/Azure*, create a database connection:

1. Click **New**.
2. Provide the following information as required:

   Database connection name: This is the name to distinguish the database connection from database connections for other database instances.

   URL: This is the URL of the XMLA Provider that was configured for HTTP access. For example: `http://`*`fully-qualified-machinename`*`/MicroStrategyMDX/MicroStrategyMDX.asmx`

   > The *`fully-qualified-machinename`* is usually of the form `machine.domain.company.com`. You can also use the IP address as the *`fully-qualified-machinename`*. The URL is not case sensitive.

   DSI: Type the fully qualified machine name or the IP address of the machine where the Microsoft Analysis Services server is running.

   Catalog: Use Microsoft's SQL Server Management Studio to view the Analysis Server which contains the cubes to work with in MicroStrategy. The database that contains the cube becomes the catalog.

Upon creating the database connection, see *Complete the Connection to Analysis Services* for next steps.

Complete the Connection to Analysis Services

After you create the database connection as described in *Create a Database Connection for Analysis Services*, complete the connection as follows:

1. Click **New**.
2. Enter a name for the database login object, a login ID, and a password. For information on allowing Windows users' login credentials to be used to execute against MDX cubes imported from Analysis Services into MicroStrategy, see *Single Sign-On to Microsoft Analysis Services, page 73*.
3. Click **OK**.
4. Choose **Default database login name** > **OK**.
5. Choose **Database connection** > **OK**.

After you have connected a database instance to your Analysis Services 2012/2014/2016/2017/2019/Azure server, you can begin accessing your Analysis Services 2012/2014/2016/2017/2019/Azure data from a MicroStrategy project. For information on configuring the MDX cube database instance in a project, see *Configuring MDX Cube Database Instances in Projects, page 66*.

Connecting to TM1 Servers

Before creating any reports using the TM1 data, you need to establish a connection to the TM1 servers.

To begin the process of connecting MicroStrategy to your TM1 servers, complete the following installations and configurations:

Install and Configure the MDX Cube Provider and TM1 OLEDB Provider

1. Install the MicroStrategy MDX Cube Provider on a machine. For information on how to install this component, which is provided with MicroStrategy, see the Installation and Configuration Help.

2. Install the TM1 OLEDB Provider on the machine where you installed MicroStrategy and the MicroStrategy MDX Cube Provider. For 32-bit operating systems, install the 32-bit TM1 OLEDB Provider, and for 64-bit operating systems, install the 64-bit TM1 OLEDB Provider.

   For information on how to install the TM1 OLEDB Provider, see your third-party TM1 documentation.

3. In the installation location for the MicroStrategy MDX Cube Provider, you must modify the `Datasources.xml` file. It is recommended that you make a backup copy before modifying this .xml file.

   - In the `Datasources.xml` file, locate the `<AuthenticationMode></AuthenticationMode>` section. Within this section, type the authentication mode used for the virtual directory running on Microsoft Internet Information Services. You can use either Basic or Integrated authentication. For information on these authentication options, refer to your third-party Microsoft documentation.

   - In the `Datasources.xml` file, locate the `<ProgramID></ProgramID>` section. Within this section, type the value `TM1OLAP`.

     The `Datasources.xml` file includes an example definition for the `<DataSources></DataSources>` section with default values for a TM1 connection. You can use this section as a template for the `<DataSources>` definition of your TM1 connection.

4. Create a database instance in MicroStrategy that connects to your TM1 server. Creating a database instance and its components requires a MicroStrategy login with administrative privileges:

   - Create a database connection with all valid TM1 connection information, including the URL of the MicroStrategy MDX Cube Provider and the TM1 server configuration.

   - Create a database login used to connect to your TM1 server.

Upon completing the installations, see *Creating a Database Instance for TM1 Servers* for next steps.

Creating a Database Instance for TM1 Servers

After you complete the installations described in *Install and Configure the MDX Cube Provider and TM1 OLEDB Provider*, create a database instance as follows:

1. In Developer, navigate to the Folder List.
2. Open your project source and expand **Administration**.
3. Choose **Configuration Managers** > **Database Instance**
4. Choose **File** > **New** > **Database Instance**.
5. In the **Database instance name** field, type a descriptive name for the database instance.
6. Choose **Database connection type** > **IBM Cognos TM1**.

After creating the database instance, see *Creating a Database Connection for TM1 Servers* for next steps.

Creating a Database Connection for TM1 Servers

After you create the database instance as described in *Creating a Database Instance for TM1 Servers*, create the database connection as follows:

1. Click **New** to create a database connection. Provide the following information as required:

   Database connection name: This is the name to distinguish the database connection from database connections for other database instances.

   URL: The URL of the MicroStrategy MDX Cube Provider that was configured for HTTP Access. This connector is installed as part of a MicroStrategy installation to support connections to TM1. The default URL is in the following format:

   `http://`*`fully-qualified-machinename`*`/MicroStrategyMDX/MicroStrategyMDX.asmx`

   The *`fully-qualified-machinename`* is usually of the form `machine.domain.company.com`. You can also use the IP address as the *`fully-qualified-machinename`*.

   DSI: The fully qualified machine name of the machine where the TM1 Admin Server is running.

   Catalog: The catalog represents the name of the TM1 server configuration, which is defined in the `TM1S.cfg` file. For more information, see your third-party TM1 documentation.

After creating the database connection, see *Complete the Connection to the TM1 Servers* for next steps.

Complete the Connection to the TM1 Servers

After you create the database connection as described in *Creating a Database Connection for TM1 Servers*, complete the connection as follows:

1. Click **New**.
2. Enter a name for the database login object, login ID, and password used to connect MicroStrategy to TM1.

3. Click **OK**.
4. Choose **Default database login name** > **OK**.
5. Choose **Database connection** > **OK**.

After you have connected a database instance to your TM1 server, you can begin accessing your TM1 data from a MicroStrategy project. For information on configuring the MDX cube database instance in a project, see *Configuring MDX Cube Database Instances in Projects, page 66*.

Configuring MDX Cube Database Instances in Projects

When a project designer creates a project in MicroStrategy, a database instance is assigned to that project. A project can only have one warehouse database instance, which is the database instance that the MicroStrategy Warehouse Catalog uses to access the warehouse tables available for the project and determine the set of relational data available to be analyzed in the project. See the Project Design Help for more information on the Warehouse Catalog, and see the Installation and Configuration Help for more information on connecting a project to a warehouse database instance.

Additional database instances connected to MDX cube sources can be included in a project along with the warehouse database instance. This enables you to connect to and report on data from your relational data warehouse as well as your MDX cube sources from within the same project. For information on importing MDX cubes with the MDX Cube Catalog, see *Importing MDX Cubes, page 83*.

MicroStrategy Administrator privileges are required when configuring MDX cube database instances.

The following topics provide details about configuring MDX cube database instances:

Removing an MDX Cube Database Instance from a Project

You can remove a database instance from a project only if there are no dependent objects in the project for the database instance. This includes removing all MDX cubes for the MDX cube source database instance that have been integrated into MicroStrategy.

To remove an MDX cube database instance from a project

1. Right-click a project and choose **Project Configuration** to open the Project Configuration Editor and view MDX cube database instances.

   MDX cube database instances display in bold text when in use, and these instances cannot be removed. For information on removing a database instance and related MDX cube managed objects from a project, refer to the System Administration Help.

2. Once all dependent objects are removed, click **Remove** from the Schema Maintenance dialog to remove an MDX cube database instance from a project.

MDX Cube Schema Loading

By default, MDX cube schemas are loaded as needed when MDX cube reports are executed. You can choose to load MDX cube schemas when Intelligence Server starts. This affects when the load time required for MDX cube schemas occurs.

Loading MDX cube schemas when the Intelligence Server starts is a good option when MDX cube reports are commonly used in a project. This optimizes the MDX cube report runtime performance since the schema for the report has already been loaded. However, The overhead experienced during Intelligence Server startup is increased due to the processing of loading MDX cube schemas, and when importing a new MDX cube or refresh an MDX cube, you must update the schema of your project to load the MDX cube before report execution and to use the updated MDX cube schema.

Loading MDX schemas when an MDX cube report is executed is a good option if MDX cube reports are supported for a project, but are rarely used in the project. This decreases the overhead experienced during Intelligence Server startup as compared to including loading MDX cube schemas as part of the startup tasks, and MDX cube schemas are not required, they do not need to be loaded and no overhead is experienced. However, the MDX cube report runtime performance can be negatively affected due to the processing of loading MDX cube schemas.

✓ This procedure assumes you have already created an MDX cube database instance that connects to your MDX cube source.

To define MDX cube schema loading options

1. In Developer, log in as an administrator to your project that is connected to an MDX cube source.
2. Right-click the project and choose **Project Configuration** to open the Project Configuration Editor.
3. Choose **Categories** > **Database instances** > **MDX Data warehouses**.
4. Click **Schema Maintenance**.
5. From the **Preload** column, choose:

   Yes: MDX cubes are loaded when Intelligence Server starts.

   or

   No: MDX cubes are loaded when MDX cube reports are executed.
6. Click **OK**.
7. Click **OK** again.
8. Whenever you modify the schema loading option for an MDX cube source database instance, you must update the schema for your project

to reflect these changes.

In **Developer** choose **Schema** > **Update Schema**.

Exchanging the Database Instance for an MDX Cube Schema

When you integrate MDX cube sources into MicroStrategy, the data is integrated as an MDX cube schema. Once you integrate an MDX cube source into MicroStrategy, you can exchange the database instance used to connect to the MDX cube schema for a different database instance. This allows you to use different database instances with different login and connection information to access an MDX cube schema.

- An MDX cube source has been integrated into MicroStrategy (see *Chapter 3, Integrating MDX Cubes into MicroStrategy*).
- You have created the database instance where the MDX cube schema should be moved. This database instance must meet the following requirements:
  - The database instance does not have any MDX cube schema defined for it.
  - The database instance uses the same database connection type as the database instance that currently stores the MDX cube schema. For example, you cannot move an MDX cube schema for SAP data to a database instance that has connection information for a Microsoft Analysis Services MDX cube source.

To exchange the database instance for an MDX cube schema

1. In Developer, log in as an administrator to your project that is connected to an MDX cube source.
2. Right-click the project and choose **Project Configuration**.
3. Choose **Categories** > **Database instances** > **MDX Data warehouses**.
4. Click **Schema Maintenance**to view a list of all available MDX cube schemas for the project.

5. From the **Database instance** column for an MDX cube schema, choose the database instance to move the MDX cube schema to.
6. Click **Yes**.
7. Click **OK** twice.

Supporting Large Result Sets for MDX Cube Reports

The maximum number of result rows allowed for an MDX cube report is enforced by the same governing setting which is set for reports that have their results returned from a relational database. However, queries on MDX cube sources during MDX cube report execution can return more result rows than may be expected, which may cause some MDX cube reports to fail when the number of rows is exceeded.

When MDX is executed against an MDX cube source, additional result sets are required because an MDX data set is not simply a flat table as in relational databases. MDX data sets need to account for ragged and unbalanced hierarchies and other customizable relationships between attribute elements that are not always possible to model in relational databases. The additional information is used to reconstitute the data set in MicroStrategy as it exists in the MDX cube source.

To support the possibility of a large number of result rows returned by MDX cube reports, you must define the governing setting to a higher maximum.

To support large result sets for MDX cube reports

1. In Developer, log in to a project that is connected to an MDX cube source.
2. Right-click the project and choose **Project Configuration**.
3. Choose **Categories** > **Governing Rules**> **Result sets**.

4. Underneath **Final Result Rows**, define the **All other reports** governing setting to a number that is high enough to support the execution of your MDX cube reports.

   The total number of result rows for an MDX cube report can be at least four times the number of final result rows displayed on an MDX cube report.

5. Click **OK**.

Defining MDX Cube Sources to Inherit Formats for Metric Values

You can inherit value formats from your MDX cube source and apply them to metric values in MicroStrategy MDX cube reports to support flexible formatting. If you do not inherit value formats, you can only apply a single format to all metric values on an MDX cube report. For an explanation of all of the reporting benefits of inheriting value formats from your MDX cube source, see *Inheriting MDX Cube Source Formats for Metric Values, page 178*.

By default, MDX cube reports with database instances that inherit MDX cube source formats will automatically inherit those formats. You can also specify whether to inherit these formats on a report-by-report basis, as described in *To inherit MDX cube source formats for metric values , page 179*.

To define MDX cube source database instances to inherit MDX cube source formatting

1. In Developer, log in with administrative privileges to a project source.
2. In the Folder List, choose **Administration** > **Configuration Managers** > **Database Instance**.
3. Right-click an MDX cube source database instance and choose **VLDB Properties**.
4. Choose **Tools** > **Show Advanced Settings**.

5. Choose **VLDB Settings** > **MDX** > **MDX Cell Formatting**.

6. Clear the **Use default inherited value** check box.

7. Choose one of the following options:

   MDX metric values are formatted per column: If you choose this option, MDX cube source formatting is not inherited. You can only apply a single format to all metric values on an MDX cube report.

   or

   MDX metric values are formatted per cell: If you choose this option, MDX cube source formatting is inherited. Metric value formats are determined by the formatting that is available in the MDX cube source, and metric values can have different formats.

8. Click **Save and Close**.

9. Restart the Intelligence Server.

Authentication

Most standard MicroStrategy platform authentication features also apply to MDX cube sources and MDX cube reports, using one of the following supported methods:

- **NT (Windows) authentication**: Uses your network login ID to authenticate a connection to MicroStrategy Intelligence Server. NT (Windows) authentication can be used to authenticate the user to the Intelligence Server, but not to MDX cube sources.

- **Standard authentication and LDAP authentication**: Are supported independently from the data source that is being used. Standard authentication and LDAP authentication can be used to authenticate the user to the Intelligence Server, but not to MDX cube sources.

- **Connection mapping**: Is supported the same way as for standard MicroStrategy reports. In addition, specific connection mappings may be designated for each database instance and user or group combination.
- **Database authentication**: Is supported in the same way as for relational data providers. If multiple sources are configured for database authentication, the same login information must be applicable to all sources.
- If your project connects to SAP BW as an MDX cube source, you can enable users to log in to a MicroStrategy project with their SAP user credentials and use SAP BW roles as a method to grant the users privileges in MicroStrategy. This SAP user creation and security in MicroStrategy requires that you use the MicroStrategy database authentication option. For information on importing SAP users and roles into MicroStrategy, see *Authenticating SAP BW Users in MicroStrategy Projects, page 76*.
- **Integrated authentication**: Is supported for single sign-on authentication to Microsoft Analysis Services and SAP BW MDX cube sources. For information about the single sign-on functionalities supported for MDX cube sources and necessary configuration steps, see *Single Sign-On to Microsoft Analysis Services* and *Single Sign-On to SAP BW*

In addition to these methods for authentication within the MicroStrategy platform, the following topics explain the authentication and permissions required for other systems:

Single Sign-On to Microsoft Analysis Services

MicroStrategy supports single sign-on authentication of a Windows user to MicroStrategy Developer, MicroStrategy Web, and Microsoft Analysis Services serving as an MDX cube source in MicroStrategy, using the integrated authentication option.

Integrated authentication enables a user to log in once to their network account, using Kerberos to validate the user's credentials, and access

MicroStrategy Web and Microsoft Analysis Services without having to log in again. With integrated authentication, Windows users' login credentials can also be used to execute against MDX cubes imported from Analysis Services into MicroStrategy. To configure this type of authentication, see the *Identifying Users: Authentication* section of the System Administration Help.

To enable Kerberos authentication pass-through to Analysis Services, you must install the MicroStrategy MDX Cube provider on the same machine with the MicroStrategy Intelligence Server.

After you have enabled integrated authentication, you must allow access to Microsoft Analysis Services MDX cubes. To enable a Windows user access to these MDX cubes in MicroStrategy, you must assign the Windows user membership to one or more Analysis Services security role definitions. For more information, see *Connecting to Analysis Services Servers, page 57*.

Single Sign-On to SAP BW

MicroStrategy supports single sign-on authentication of a Windows user to MicroStrategy Developer, MicroStrategy Web, and SAP BW serving as an MDX cube source in MicroStrategy using the integrated authentication option.

Integrated authentication enables a user to log in once to their network account, confirming access to the SAP BW data that has been integrated into MicroStrategy.

Integrated authentication enables a user to log in once to their network account. This can include the user's initial log in to their computer. This same authentication is used to confirm access to the SAP BW data that has been integrated into MicroStrategy. To support this type of authentication, you must complete the following requirements:

- Configure the SAP NetWeaver Application Server Java to use your SAP BW server as the repository for your SAP user accounts, accept Kerberos tickets as a valid form of login credentials, and generate an associated

SAP Logon Ticket.

- You must include an additional connection parameter, `EPURL`, as part of the database instance that you created for your SAP BW connection. See *Add the EPURL Connection Parameter to Your Database Instance* for steps to include this parameter.

For additional information about configuring this type of authentication, see the *Identifying Users: Authentication* section of the System Administration Help.

Add the EPURL Connection Parameter to Your Database Instance

1. Within MicroStrategy Developer, right-click the database instance for your SAP BW connection and choose **Edit**.
2. In the Database connection area, choose the database connection and click **Modify**.
3. On the **Advanced** tab, add the `EPURL` parameter to the Connection Settings. The EPURL should be formatted like:

   `EPURL=http://`*`Server`*`:`*`Port`*`/irj/portal;`

   Where:

   `Server` is the host name of the Java version of your SAP server. This must match how the service principal name is defined in your keytab file used for configuring your server with Kerberos. For example, you must include the domain if the domain is provided before the `@` symbol for the service principal name.

   and

   `Port` is the port number used to communication to the Java version of your SAP server.

4. Click **OK** twice.

Authenticating SAP BW Users in MicroStrategy Projects

By importing SAP BW users and SAP BW roles into MicroStrategy, you can enable users to log in to a MicroStrategy project that is connected to an SAP BW MDX cube source with their SAP user credentials, and use SAP BW roles as a method to grant the users privileges in MicroStrategy.

Imported SAP BW users are granted group privileges at runtime. This means that when a user attempts to perform a task that requires a privilege, it is at this time that the group membership is checked and privileges are granted. This is visible in the User Editor: A MicroStrategy user (associated with an SAP BW user) does not list the privileges it inherits from its MicroStrategy groups (associated with SAP BW roles). Also, users are not shown as members of the group.

SAP BW users and roles are imported into MicroStrategy when a user logs in to a MicroStrategy project with their SAP BW user name and password.

You must establish a connection between MicroStrategy and the SAP BW system using a MicroStrategy database instance. Steps for this process are provided in *Connecting to SAP BW servers , page 43*.

Complete the following to authenticate SAP BW users in MicroStrategy projects

1. *Define SAP BW as an Authentication Database*
2. *Grant SAP BW Users Access to MicroStrategy Developer and Web*
3. *Enable SAP BW User and Group Import Options*
4. *Define Project Sources to Authenticate Using SAP BW Credentials*

Define SAP BW as an Authentication Database

To use SAP BW user credentials to log in to a MicroStrategy project, you must define your SAP BW MDX cube source as an authentication database for the project. This means that your SAP BW MDX cube source is the storage location for users and their respective credentials.

Define your SAP BW database instance as the authentication database instance for the MicroStrategy project that contains your MDX cubes.

1. In MicroStrategy Developer, log in to a project source with a project connected to an MDX cube source.
2. From the **Folder List**, right-click a project and choose **Project Configuration**.
3. From the **Categories** list, choose **Database instances** > **Authentication** > **Metadata**.
4. From the Database Instances Authentication User drop-down, choose the database instance that connects to your SAP BW MDX cube source.
5. Click **OK**.

Grant SAP BW Users Access to MicroStrategy Developer and Web

Grant the Use Developer and Web User privileges to the Warehouse Users group for the project that is connected to your MDX cube source.

SAP BW users and roles are imported into MicroStrategy as users and groups when an SAP BW user logs in to a MicroStrategy project. For an SAP BW user to log in to a MicroStrategy project and be imported, you must grant the Use Developer or Web User privileges to the Public/Guest group in MicroStrategy.

We recommend granting a minimum number of privileges to the Public/Guest group, which enables users to attempt to log in to a MicroStrategy project with their SAP BW user credentials. This attempted login imports the SAP BW user and roles into MicroStrategy as determined by the options described in *Authenticating SAP BW Users in MicroStrategy Projects*.

With only these privileges applied to the Public/Guest group, the user is denied access to the project. This enables the user and any associated groups to be imported before you grant specific, additional privileges to the user for the project. You can then grant the appropriate privileges directly to the imported user and any associated groups, rather than granting such privileges to the Public/Guest group.

To create a security role

✓ To grant SAP BW users access to MicroStrategy Developer and Web, log in to MicroStrategy Developer, to a project source using an account with administrative privileges.

You can use any security role that has the Use Developer and Web User privileges selected. This procedure demonstrates how to create a security role with only the Use Developer and Web User privileges.

1. From the Folder list, choose **Administration** > **Configuration Managers** > **Security Roles** > **New** > **Security Roles**.
2. In the **Name** field, type **Use Developer and Webuser**.
3. Choose **Privileges** > **Available privileges** > **Web Reporter**, and select the **Web user** check box.
4. Choose **Privileges** > **Available privileges** > **Analyst**, and then select the **Use Developer** check box.
5. Click **OK**.

To assign a security role to the Public/Guest group

1. Choose **Folder List** > **Administration** > **User Manager**.
2. Right-click the **Public/Guest** group, and choose **Edit**.
3. On the **Project Access** tab, in the **Security Role Selection** row at the top, click the drop-down list for the project connected to an SAP BW

MDX cube source, and choose **Use Developer and Web user + Inherited Access**.

4. Click **OK**.

Enable SAP BW User and Group Import Options

SAP BW users and roles are imported into MicroStrategy when a user logs in to a MicroStrategy project with their SAP BW user name and password. How SAP BW users and roles are imported into MicroStrategy is determined by options available in the MicroStrategy Intelligence Server Configuration Editor.

To enable SAP BW user and group import options

1. In Developer, log in to a project source using an account with administrative privileges.
2. Choose **Administration** > **Server**> **Configure MicroStrategy Intelligence Server**.
3. Choose **Categories** > **SAP User management**.
4. Choose the check boxes for the following options:

   Import users: A user with the same SAP BW username is created in MicroStrategy, within the Warehouse Users group.

   and

   Search for groups: The MicroStrategy groups assigned to a MicroStrategy user are synchronized with the SAP BW roles assigned to the SAP BW user. This means that if SAP BW roles have been added or removed for an SAP BW user, the associated MicroStrategy user is added or removed from the MicroStrategy groups that represent the SAP BW roles.

Copyright © 2024 All Rights Reserved

If you select to search for groups, the **Import Groups** option displays, allowing you to import SAP BW roles as groups in MicroStrategy.

All SAP BW roles that the SAP BW user is a member of are imported as groups into MicroStrategy. These groups are created within the Warehouse Users group and only have inherited privileges from the Warehouse Users group. Once these groups are created in MicroStrategy, you can assign privileges to these groups.

5. Click **OK**.

Define Project Sources to Authenticate Using SAP BW Credentials

Defining Project Sources to Authenticate Using SAP BW User Credentials

You must use database authentication to enable SAP BW users to log in to a MicroStrategy project with their SAP BW user credentials. Defining database authentication for a project source makes database authentication available for the projects within the project source.

To Define Project Sources in MicroStrategy Developer to Authenticate Using SAP BW User Credentials

1. In Developer, log in to a project source using an account with administrative privileges.
2. Right-click the project source and choose **Modify Project Source**.
3. On the **Advanced** tab, in the Authentication mode area, choose **Use login id and password entered by the user for Warehouse (database authentication)**.
4. Click **OK**.
5. A warning message displays that your connection to the project source will be closed. Click **Yes**.

To Define Project Sources in MicroStrategy Web to Authenticate Using SAP BW User Credentials

1. From the Windows **Start** menu, choose **Programs** > **MicroStrategy Tools** > **Web Administrator** to open the MicroStrategy Administrator Web page in a browser.
2. Choose **Intelligence Servers** > **Default Properties**.
3. In the **Login** area, to the left of the **Database Authentication** login mode, select the **Enabled** check box.

   To enable database authentication as the default mode of authentication used to log in to MicroStrategy project, choose the **Default** option.

4. Click **Save**.

INTEGRATING MDX CUBES INTO MICROSTRATEGY

Once you understand the relationships between the objects in an MDX cube source and MicroStrategy, and you connect to your MDX cube source, you can start integrating your MDX cube data into MicroStrategy.

The best place to start is with the MDX Cube Catalog, where you can perform the following tasks which are covered in this section:

Throughout these topics, SAP BW is used as the example MDX cube source but the procedures are similar for Analysis Services, Oracle Essbase, and TM1.

The importing and mapping tasks integrate your MDX cube source data into your MicroStrategy project, and therefore must be done before you can begin creating any MDX cube reports. The MDX Cube Catalog can be accessed from the **Schema** menu in MicroStrategy Developer and is available only after an MDX cube source database instance has been created. To learn how to create a database instance for an MDX cube source, see *Chapter , Connecting to MDX Cube Sources*.

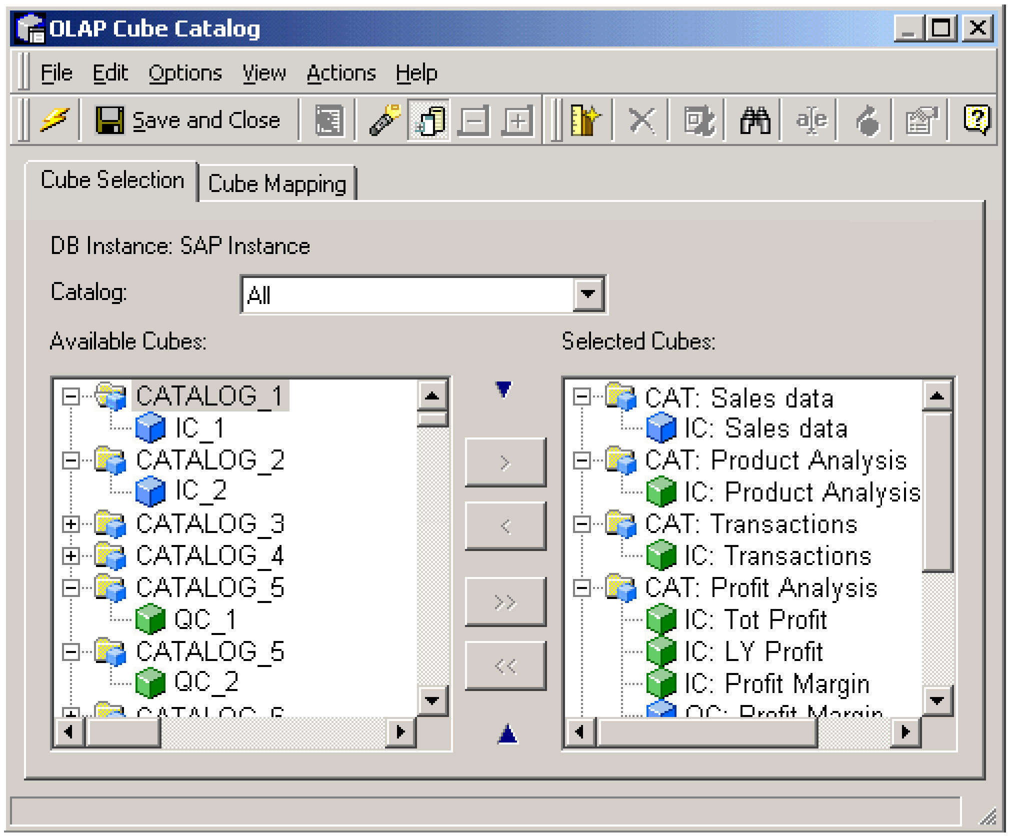
After you have fully integrated your MDX cube data into MicroStrategy, report designers can create MicroStrategy MDX cube reports, and analysts can then view and analyze data from MDX cube sources. MDX cube reports are covered in *Chapter , Reporting on MDX Cubes*.

Importing MDX Cubes

After you have connected to an MDX cube source, importing MDX cubes is the next step in integrating your MDX cube source data into MicroStrategy. Importing MDX cubes into MicroStrategy is described in the following sections:

Importing MDX cubes is performed on the Cube Selection tab of the MDX Cube Catalog. When you open the MDX Cube Catalog, all the MDX cubes are displayed under their respective catalog names in the Available Cubes pane. Using the plus (+) or minus (-) sign next to a catalog name, you can expand or hide the cubes contained in this catalog.

A catalog is designated with an icon showing a folder with a small cube super-imposed on it. An InfoCube is designated with a cube icon in blue. A query cube is designated with a cube icon in green.



DB Instance: SAP Instance

Catalog: All

Available Cubes:

- CATALOG_1
  - IC_1
- CATALOG_2
  - IC_2
- CATALOG_3
- CATALOG_4
- CATALOG_5
  - QC_1
- CATALOG_5
  - QC_2

Selected Cubes:

- CAT: Sales data
  - IC: Sales data
- CAT: Product Analysis
  - IC: Product Analysis
- CAT: Transactions
  - IC: Transactions
- CAT: Profit Analysis
  - IC: Tot Profit
  - IC: LY Profit
  - IC: Profit Margin

If you create new cubes in Analysis Services and the cubes are not displayed in the MDX Cube Catalog, you may have to modify some permissions in Analysis Services.

Importing MDX Cubes Before Report Creation

Before you can create MDX cube reports, you need to import MDX cubes from your MDX cube source into MicroStrategy.

MDX cubes can only be imported into a MicroStrategy project by users with the Import MDX Cube privilege.

You can import MDX cubes only after an MDX cube source database instance has been created. See *Chapter , Connecting to MDX Cube Sources* for more information.

If you are import MDX cubes from SAP BW, any existing SAP BW query can be released for analysis within MicroStrategy via the Query Analyzer interface, in the Query Analyzer interface dialog, and choosing **Extended** > **Allow External Access to This Query**.

If you are importing SAP BW cubes that include variables of the type Replacement Path, you must remove them before importing the cubes into MicroStrategy.

Once imported, MicroStrategy automatically maps attributes, metrics, prompts, and other objects to the MDX cube (see *Mapping MDX Cubes, page 99*). However, if you plan to create relationships between your MDX cube source data to data from a different data source included in the MicroStrategy project, you can map MDX cube data to existing attributes in the project (see *Mapping MDX Cube Data to Project Attributes, page 106*). Once the data is mapped to MicroStrategy objects, you can build reports that access the imported MDX cubes.

If the MDX cube needs to be updated to retrieve new data, or access the data in a different MDX cube data source, see *Updating MDX Cube Structure, page 94*.

To import MDX cubes

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.

2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database**

Instance drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.

3. Click **Cube Selection**.

4. From the **Catalog**, select the MDX cube to import. You can also select **All** to display the MDX cubes for all catalogs.

   To search for a specific MDX cube to import, from the **Edit** menu, select **Find**, or click the **Find** icon on the toolbar.

5. Click the plus **+** sign to expand the catalog folder.

6. Right-click the MDX cube and choose **Cube Structure** to preview the structure of an MDX cube before importing it into MicroStrategy.

7. Click **Update Structure** to synchronize with the most recent definition of MDX cube structures in the MDX cube source. This is helpful when a new characteristic or key figure has been added to an InfoCube in SAP BW.

8. In the MDX Cube Catalog, choose **Options** > **Import options**.

9. Additionally, for SAP data sources, you can determine how the two levels of a stand-alone characteristic are imported into MicroStrategy by selecting:

   Do not import Level 00 for flat hierarchies: When selected, the first level aggregate data for a stand-alone characteristic is not imported into MicroStrategy. Only the detail data level for the characteristic is imported and mapped to an attribute in MicroStrategy.

   and

   Suppress Level 01 suffix for flat hierarchies: When selected, the suffix Level 01 is not included in the attribute name mapped to the detail data level of the characteristic.

For more information on stand-alone characteristics and how these options import them into MicroStrategy, see *Importing Levels and Suffixes for Characteristics, page 88*.

10. Select the **Synchronize logical object names with source** check box if you want to synchronize the names of schema objects in MicroStrategy with the names of objects in the MDX cube source.

    If you select this check box, the attributes, hierarchies, metrics, and other schema objects in MicroStrategy are renamed if the names in the MDX cube source are updated. This name synchronization is applied when you update the MDX cube structure.

11. Additionally, for Essbase and TM1 data sources, you can select the **Import measure as a regular dimension** check box if you want to import additional measure structure from MDX cube sources.

    If the check box is greyed out and unavailable for selection, this means the MDX cube source you are connected to cannot support the integration of additional measure structure.

    If you select this check box, the additional measure structure is integrated into MicroStrategy, and can support the hierarchal display of measures in an MDX cube source. See *About Importing an Additional Measure Structure, page 90* for examples.

12. Click **OK**.
13. Select the MDX cubes to import and click the single arrow **>**. To import all the MDX cubes, click the double arrows **>>**.
14. To remove an MDX cube, right-click any MDX cube in the Selected Cubes pane and choose **Remove [*cube name*]**.
15. Click **Save**.

Once the first MDX cube for an MDX cube source is imported into MicroStrategy, a Data Explorer is added to the MicroStrategy project. The

Data Explorer helps you browse through data for its associated MDX cube source. You can find the Data Explorer for the MDX cube source in the Folder List of Developer, under the associated MicroStrategy project.

Importing Levels and Suffixes for Characteristics

When characteristics in MDX cubes are imported into MicroStrategy, they can be imported both as a stand-alone characteristic and as part of a hierarchy defined in the MDX cube source. See *To import MDX cubes, page 85* for information about selecting these options as part of the MDX cube import process.

Stand-alone characteristics are imported into MicroStrategy with two levels: the first level is an aggregate of all the characteristic data, and the second level is the detail data. You may choose not to import the aggregate level of a characteristic into MicroStrategy.

You can determine how the two levels of a stand-alone characteristic are imported into MicroStrategy for the first time:

- **Do not import Level 00 for flat hierarchies**: Choose this check box to exclude the first level aggregate data for a stand-alone characteristic during the import into MicroStrategy. Only the detail data level for the characteristic is imported and mapped to an attribute in MicroStrategy.
- **Suppress Level 01 suffix for flat hierarchies**: Choose this check box to exclude the suffix Level 01 from the attribute name mapped to the detail data level of the characteristic. An attribute named Region (rather than Region Level 01) is mapped to the detail data of the imported Region characteristic.

Attributes can be shared by multiple MDX cubes imported into a MicroStrategy project, affecting the import behavior.

If you select both import options for a Region characteristic, the characteristic is imported into MicroStrategy so that only the detail level is imported and mapped to an attribute, and the attribute name does not include the suffix Level 01.

These options do not affect characteristics when they are imported as part of a hierarchy. All levels and suffixes for the hierarchy are imported and mapped into MicroStrategy regardless of whether you selected any of the import options.

If you plan to map your MDX cube data to project attributes that are part of a relational schema, you should map the stand-alone characteristics to the project attributes rather than the levels of a hierarchy. For more information on mapping MDX cube data to project attributes, see *Mapping MDX Cube Data to Project Attributes, page 106* and *Mapping MDX Cube Data to Project Attributes, page 106*.

Shared attribute effects on import behavior

MDX cube characteristics mapped to attributes in MicroStrategy can be shared by multiple MDX cubes, which can affect how and when attributes are imported for MDX cubes.

The import options to exclude the Level 00 attributes and Level 01 suffixes are described in *Importing Levels and Suffixes for Characteristics*. The descriptions assume that the characteristics for the MDX cube are being imported into MicroStrategy for the first time. By default, Level 00 attributes are not created and Level 01 suffixes are excluded from the attribute names mapped to Level 01 data.

After importing this MDX cube and Region characteristic, you can import another MDX cube that shares the region data. If you use the same import options, the MDX cube re-uses the attribute already created for the characteristic. However, modifying the import options has the following impacts on import behavior:

- Level 00 data is imported and a Level 00 attribute is created if you choose to import Level 00 data. A Level 00 attribute is created and shared for all MDX cubes that share imported attributes for the characteristic. This includes MDX cubes that were imported previously and were set to exclude Level 00 data.

  Once a Level 00 attribute has been imported into MicroStrategy, all MDX cubes that share this data must include this attribute, regardless of whether you choose to include Level 00 data or not.

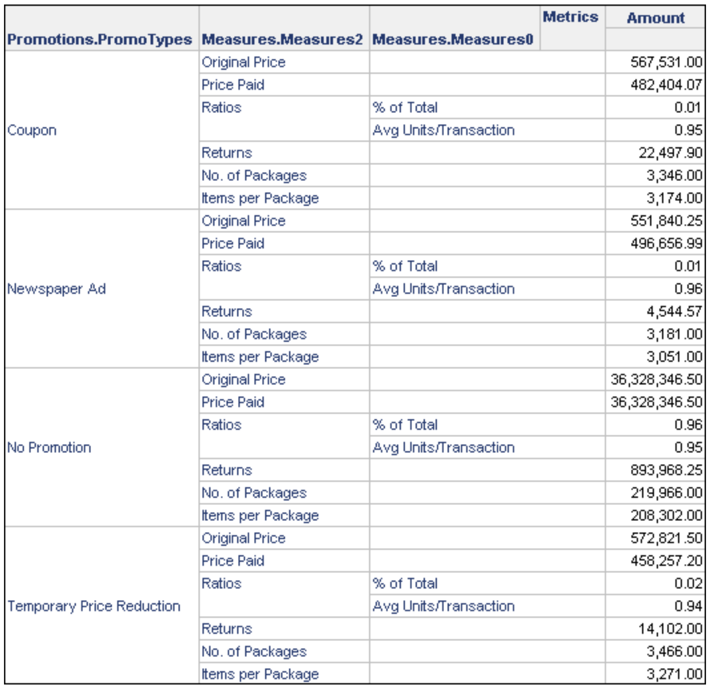
- The Level 01 suffixes are not included even if you choose to include Level 01 suffixes. This is to maintain a consistent schema across all MDX cubes that share data.

  This is also the case if on your first import you choose to include Level 01 suffixes. The suffixes are included for any imported attributes. If you then choose to exclude the Level 01 suffixes, the suffixes are still included for any attributes that have already been imported and are shared by multiple MDX cubes.

  You can rename an attribute from within the MDX Cube Catalog or MDX Cube Editor at any time. These modifications are reflected in all MDX cubes that share the modified attribute.

About Importing an Additional Measure Structure

Measures in MDX cube sources are integrated into MicroStrategy as metrics by default. However, measures can include an additional structure that cannot be supported by MicroStrategy metrics. This additional structure can support the hierarchical display of measures in an MDX cube source, and the Measures attributes display the measures for the MDX cube, including the hierarchical structure of the measure.



		Metrics	Amount
Promotions.PromoTypes	Measures.Measures2	Measures.Measures0	
Coupon	Original Price		567,531.00
	Price Paid		482,404.07
	Ratios	% of Total	0.01
		Avg Units/Transaction	0.95
	Returns		22,497.90
	No. of Packages		3,346.00
	Items per Package		3,174.00
Newspaper Ad	Original Price		551,840.25
	Price Paid		496,656.99
	Ratios	% of Total	0.01
		Avg Units/Transaction	0.96
	Returns		4,544.57
	No. of Packages		3,181.00
	Items per Package		3,051.00
No Promotion	Original Price		36,328,346.50
	Price Paid		36,328,346.50
	Ratios	% of Total	0.96
		Avg Units/Transaction	0.95
	Returns		893,968.25
	No. of Packages		219,966.00
	Items per Package		208,302.00
Temporary Price Reduction	Original Price		572,821.50
	Price Paid		458,257.20
	Ratios	% of Total	0.02
		Avg Units/Transaction	0.94
	Returns		14,102.00
	No. of Packages		3,466.00
	Items per Package		3,271.00

To support this additional structure when importing the data into MicroStrategy, you can import the measures as a regular MDX dimension. This method integrates the measures into MicroStrategy as an attribute, which can support the additional structure for the measures. See *Importing an Additional Measure Structure from MDX Cube Sources* for steps.

To support this additional structure in measures, be aware that:

- The additional structure for measures can only be integrated into MicroStrategy for Oracle Essbase MDX cube sources. This option is not available for all other MDX cube sources.
- You must choose to support this additional structure when importing the MDX cube into MicroStrategy. If the MDX cube is already imported without this support, you must remove the MDX cube and import it into MicroStrategy again.
- If you enable support for the additional structure, you cannot create compound metrics for the MDX cube. Creating compound metrics for MDX cubes is described in Creating Metrics from MDX Cube Data.

- If you enable support for the additional structure, a single metric called Amount is created for the MDX cube. When using this Amount metric, be aware of the following:
  - You must include the Amount metric, as well as the attributes created for the measures, on an MDX cube report to display the values for all the measure data. The attributes for the measures are created under a hierarchy named Measures by default. The example report displays an example of the Amount metric and Measures attributes on a report.
  - Since the measure data is displayed using a single Amount metric, only a single value format such as currency or percentage can be used for all the values. The example report uses a fixed value format that displays two decimal places.
  - To support multiple value formats, you must disable the support for this additional structure, which imports each measure as a separate metric in MicroStrategy.
  - Metric qualifications are not supported for the Amount metric. If you create a metric qualification for the Amount metric, it can cause an error when the MDX cube report is executed.

Importing an Additional Measure Structure from MDX Cube Sources

See *About Importing an Additional Measure Structure* for details and special considerations when importing additional measure structures from MDX cube sources.

MDX cubes can be imported into a MicroStrategy project only by an architect with the Import MDX Cube privilege.

You can import MDX cubes only after an MDX cube source database instance has been created. For information on creating an MDX cube source database instance and connecting to an MDX cubes source, see *Chapter , Connecting to MDX Cube Sources*.

1. In MicroStrategy Developer, log in to a project that is connected to a Oracle Essbase MDX cube source. Only one user can edit a project at a time.
2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.
3. In the MDX Cube Catalog, from the **Options** menu, choose **Import options**.
4. Choose the **Import measure as a regular dimension** check box.

   If the check box is grayed out and unavailable for selection, this means that the MDX cube source you are connected to cannot support the integration of additional measure structure. The additional structure for measures can only be integrated into MicroStrategy for Oracle Essbase MDX cube sources.
5. Click **OK**.
6. Click **Cube Selection**.
7. From the **Catalog** drop-down list, select the MDX cube to import, or select **All** to display the MDX cubes for all catalogs.

   Select **All** to display the MDX cubes for all catalogs, or search for a specific MDX cube to import by choosing **Edit** > **Find**.
8. Click the plus **+** sign to expand the catalog folder.
9. Select the MDX cubes to import and click the single arrow **>**. To import all the MDX cubes, click the double arrows **>>**.
10. Click **Save**.

11. Click **Cube Mapping**.

12. From the **Catalog\Cube** drop-down list, select the MDX cube you just imported. You can view the new objects created to support the additional measure structure:

    Measures dimension: In the Physical view column, the Measures dimension includes attributes that represent the additional structure of the measures as they exist in the MDX cube source.

    Measures: In the Physical view column, the Measures object displayed at the bottom of the list includes a single Amount metric.

13. Continue to map the data in the MDX cube to prepare the data for reporting in MicroStrategy (see *Mapping MDX Cubes, page 99*).

Importing MDX Cubes During Report Creation

When you create an MDX cube report, you choose MDX cubes for your report from the Select Cube dialog box which can also be used to import cubes by using the **Retrieve cubes** option. This option is available only to users with the Import MDX Cubes privilege and must be done after a database instance has been defined. See *Chapter , Connecting to MDX Cube Sources* for more information about each cube source.

⚠ You can only import and map a single MDX cube when importing an MDX cube for your MDX cube report. This method also does not have all the options to map the MDX cube data to MicroStrategy objects. The MDX cube is imported with all objects mapped to the default managed objects (see *About Managed Objects, page 97*).

You can search for an MDX cube for your report by clicking the **Find** button at the bottom of the Select Cube dialog box.

Updating MDX Cube Structure

Updating the structure of an MDX cube synchronizes the MDX cube definition in the MicroStrategy project with the latest MDX cube model in the

MDX cube source. As a result, any addition or deletion of levels is reflected in the MDX cube structure that has been imported.

You can update an MDX cube to point to a different MDX cube from the same MDX cube source. For example, your current MDX cube includes data for the year 2014. A new source of data for 2015 is to be included in MicroStrategy as an MDX cube. You can import this data as a new MDX cube in MicroStrategy. Alternatively, you can update the current MDX cube to point to this new data. This update requires that the data for both MDX cubes is similar in structure so that the definition of the MDX cube in MicroStrategy is maintained. You can use Command Manager to apply this update to an MDX cube, and the scripts required are described in the help for Command Manager Help.

If any MDX cubes have been deleted from the cube source, that information is also updated in MicroStrategy. If any MicroStrategy reports used those MDX cubes, those reports will fail when they are run again.

If you migrate MDX cube reports and other MDX cube objects between multiple projects, all MDX cube updates should be performed in a single project and migrated over to the other projects. For best practices on maintaining MDX cubes between multiple projects, see *Maintaining MDX Cubes Between Multiple Projects, page 96*.

To update an MDX cubes structure

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.

2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.

3. Choose **Cube Selection**.
4. In the **Selected Cubes** pane, right-click an MDX cube and select **Update Structure**. If a new dimension has been added to a cube, new attributes and metrics are automatically mapped in MicroStrategy and displayed on the Cube Mapping tab to reflect the change.

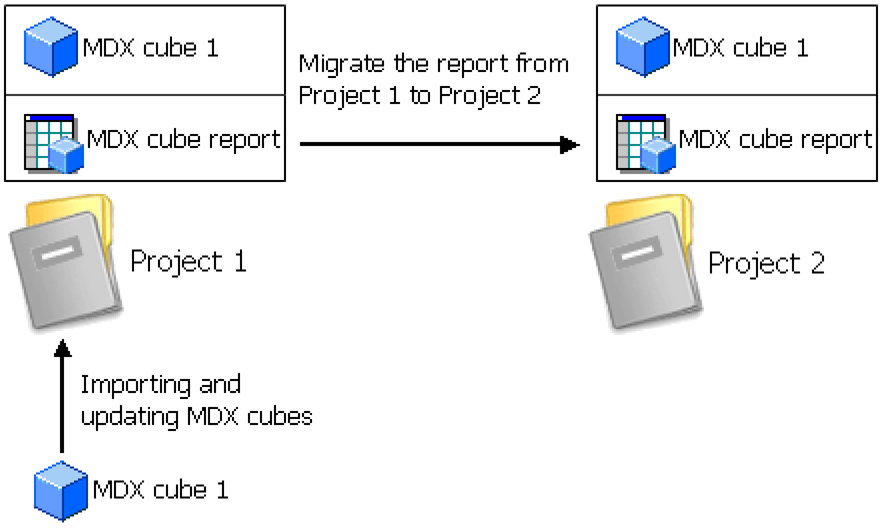
   You can also modify any new MDX cube object mappings to MicroStrategy objects in the Cube Mapping tab.
5. Click **Save and Close**.

Maintaining MDX Cubes Between Multiple Projects

When developing a reporting environment for MDX cube data, a common practice is to use both an initial testing project, where MDX cubes are imported and MDX cube reports are created, and a production project, where the reports are then migrated for access by users who are reporting and analyzing the MDX cube data.

If you use this multiple project scenario to develop and maintain your reporting environment for MDX cube data, we recommend the following best practices:

- To maintain object consistency between the two projects, MDX cubes should be imported and updated only in the testing project, where the MDX cube reports are created. These MDX cubes are then included in the production project as part of the process to migrate the MDX cube reports from the testing project to the production project.


MDX cube 1
MDX cube report
Migrate the report from Project 1 to Project 2
MDX cube 1
MDX cube report
Project 1
Project 2
Importing and updating MDX cubes
MDX cube 1

You can use Object Manager to migrate MDX cube reports between projects. See the System Administration Help for more information.

- If the data for an MDX cube needs to be refreshed in the production project, this MDX cube update should be performed in the testing project before migrating to the production project. This ensures that the underlying objects remain consistent across projects.
- To reduce the chance that MDX cubes are imported or refreshed in the production project, the Import MDX cube privilege should be restricted in the production project.

About Managed Objects

In MicroStrategy, standard schema objects relate the information in the logical data model and physical warehouse schema to the MicroStrategy environment. Managed objects are a type of schema object that relate MDX cube source data to the MicroStrategy environment. When an MDX cube is imported into a MicroStrategy project, managed objects (attributes, metrics, columns, tables, and so on) are created to describe the MDX cube. These schema objects are created automatically by MicroStrategy, which allows an MDX cube to be integrated quickly into your MicroStrategy project.

If your project contains both MDX cube source data and data mapped to a separate relational data source, managed objects do not allow you to create a direct relationship between the two sources of data. To solve this, you must map your MDX cube source data to project attributes that are part of your relational data model.

A managed object is just like a standard MicroStrategy object except that it is created automatically by the system and is stored in a special system folder that is hidden from users.

To Access Managed Objects

1. In MicroStrategy Developer, log in to the project that contains the managed objects you are searching for.
2. Right-click the project and choose **Search for Objects**.
3. Choose **Tools** > **Options**.
4. Select the **Display Managed Objects** check box and click **OK**.

   You can have the search return only managed objects by choosing **Display Managed Objects Only**.

5. Enter any other search criteria to meet your search requirements and click **Find Now**.
6. Once the managed objects are listed in the search result, you can rename or edit a managed object by right-clicking its name.

A managed object can be removed once it is no longer referenced by another object in the project. The removal of unused managed objects is usually performed by an administrator. For more information on removing a database instance and its related managed objects, see the *Managing Your Projects* section of the System Administration Help.

Mapping MDX Cubes

Mapping MDX cubes to objects in MicroStrategy is described in the following sections:

About Mapping MDX Cubes

When a MicroStrategy architect defines a project, much of the process centers on identifying logical entities, such as attributes and facts, that exist in physical tables. For example, an architect might identify that the key for the Customer attribute exists in the table `LU_CUSTOMER`. Once the logical entities are identified, the architect can then define a logical and physical model in the MicroStrategy metadata. This model is referenced by the MicroStrategy SQL Engine to generate SQL when a user executes a report.

In the context of MDX cube sources, an MDX cube, instead of a single table, contains all the metadata information necessary to define a logical model and physical model. When you, as the architect, need to add an MDX cube to a project in MicroStrategy, you can choose an MDX cube by using the MDX Cube Catalog or Select Cube dialog box, as described in *Importing MDX Cubes, page 83*.

When an MDX cube is imported into MicroStrategy, by default, a MicroStrategy MDX cube is created that maps to the definition of the source cube in the MDX cube source. Intelligence Server automatically creates new attributes, metrics, hierarchies, and other objects that reflect the data and levels of the imported MDX cube. Although these objects, referred to as managed objects, are part of the project, they are not related to the existing project schema and schema objects.

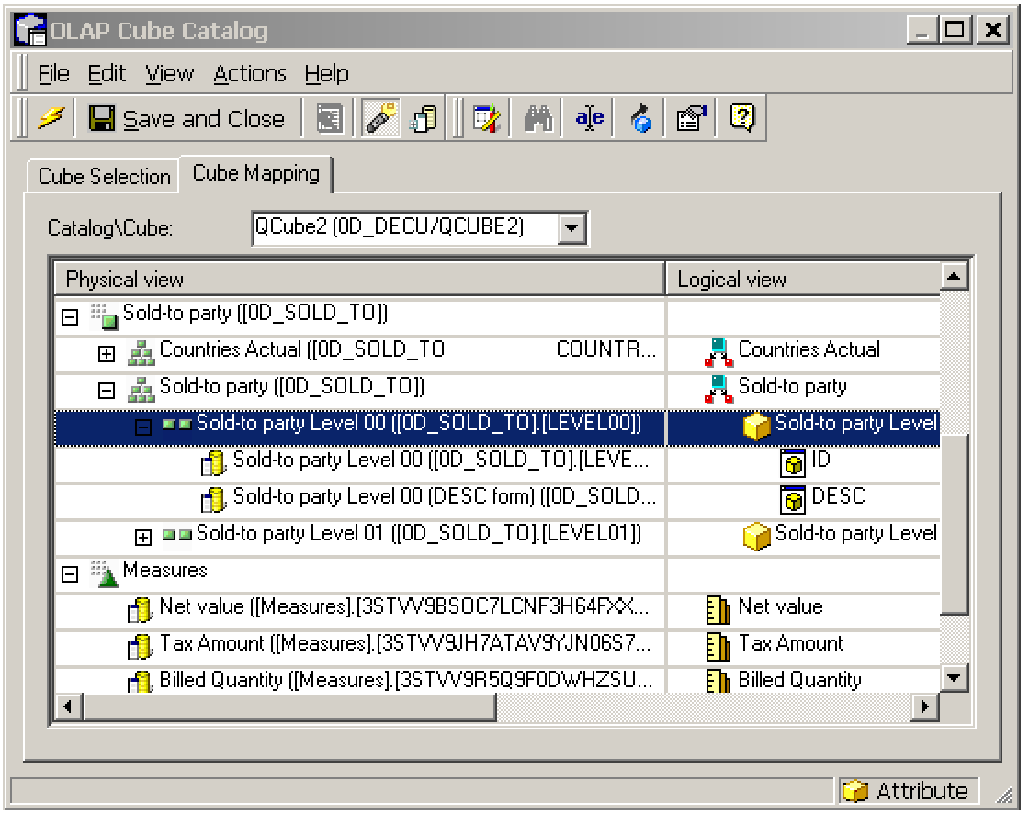
For example, within a given project, a new managed object named Year has no relation to a Year attribute that is mapped to relational data. A new schema is created for each MDX cube source database instance used in a MicroStrategy project. If you plan to use your MDX cube source as its own system of data which does not relate to any other data sources within the

MicroStrategy project, managed objects provide a quick way to integrate your data into MicroStrategy.

However, if you plan to create relationships between your MDX cube source data to data from a different data source included in the MicroStrategy project, you can map MDX cube data to existing attributes in the project. This allows data to be joined across sources in Report Services documents, which ensures that a consistent logical model is maintained. Mapping MDX cube data to existing attributes can also facilitate the use of MicroStrategy features such as security filters.

All MDX cube mapping tasks can be completed using the Cube Mapping tab in the MDX Cube Catalog.

After you have imported an MDX cube, you can perform the same mapping tasks available in the Cube Mapping tab of the MDX Cube Catalog by editing the MDX cube with the MDX Cube Editor. To use the MDX Cube Editor, right-click an MDX cube in Developer and select **Edit**.



Catalog\Cube: QCube2 (0D_DECU/QCUBE2)

Physical view		Logical view
⊟ Sold-to party ([0D_SOLD_TO])		
⊞ Countries Actual ([0D_SOLD_TO	COUNTR...	Countries Actual
⊟ Sold-to party ([0D_SOLD_TO])		Sold-to party
⊟ Sold-to party Level 00 ([0D_SOLD_TO].[LEVEL00])		Sold-to party Level
Sold-to party Level 00 ([0D_SOLD_TO].[LEVE...		ID
Sold-to party Level 00 (DESC form) ([0D_SOLD...		DESC
⊞ Sold-to party Level 01 ([0D_SOLD_TO].[LEVEL01])		Sold-to party Level
⊟ Measures		
Net value ([Measures].[3STVV9BSOC7LCNF3H64FXX...		Net value
Tax Amount ([Measures].[3STVV9JH7ATAV9YJN06S7...		Tax Amount
Billed Quantity ([Measures].[3STVV9R5Q9F0DWHZSU...		Billed Quantity

Once an MDX cube is imported into MicroStrategy, you can use MicroStrategy's Data Mining Services features to perform predictive analysis on your MDX cube data. For steps on how to include predictive analysis with your MDX cube data, see the Advanced Reporting Help.

After the MDX cube is mapped, it can be used to build reports and documents in MicroStrategy. For information on creating MDX cube reports, see *Chapter , Reporting on MDX Cubes*.

MDX Cube Mapping Requirements and Techniques

Mapping MDX cubes in MicroStrategy includes mapping and configuring objects in MicroStrategy as well as supporting various objects and data structures from MDX cube sources within MicroStrategy. Before you can perform any of the MDX cube mapping requirements and techniques, you need to import MDX cubes from your MDX cube source into MicroStrategy.

Below is a list of MDX cube mapping requirements as well as various MDX cube mapping techniques you can perform (with references to sections for further instruction) from the Cube Mapping tab of the MDX Cube Catalog:

An ID form must be mapped for each attribute.

MicroStrategy uses the ID form to map to the primary ID column for an attribute, which contains attribute element identification codes.

Only the ID and DESC forms are displayed and automatically mapped by default

By default, only the ID and DESC forms are displayed and automatically mapped for each attribute. MicroStrategy uses the DESC form to map to the primary description column for an attribute, which commonly contains descriptive information for the attribute. From the **View** menu, select **Display All Columns** to display or hide the additional attribute forms. Once displayed, you can then map these additional forms as required.

Source objects can be manipulated depending on how their related objects are defined

The MDX cube structure within the MDX cube source is represented in the **Physical View** left-hand column. This column is primarily to display the structure of the MDX cube data within the MDX cube source, as most manipulations can only be made on the related MicroStrategy objects. However, there are MDX cube source objects that can be manipulated in ways that affect how their related MicroStrategy objects are defined:

- Dimensions in MDX cube sources are not directly mapped to MicroStrategy objects. However, the levels within a dimension in an MDX cube source are mapped to hierarchies and attributes in MicroStrategy.

  The MDX cube source dimension is also where you define whether to preserve the order of MDX cube source data mapped to attribute elements in MicroStrategy. See *Preserving Attribute Element Orders from MDX Cube Sources, page 117* for details.

- MDX cube source objects mapped to MicroStrategy hierarchies can be manipulated in the following ways:

  Unbalanced or ragged hierarchies: MicroStrategy's data modeling conventions do not support unbalanced or ragged hierarchies. However, your MDX cube source may support and contain unbalanced or ragged hierarchies. To support these types of hierarchies in MicroStrategy, you must define these hierarchies as unbalanced or ragged. See *Defining Unbalanced and Ragged Hierarchies, page 122* for details.

  Including hierarchies directly on MDX cube reports: Report designers can include MicroStrategy hierarchies directly on MDX cube reports. The set of attributes that are displayed in place of a hierarchy on an MDX cube report by default is defined for MDX cube source objects mapped to MicroStrategy hierarchies. See *Displaying Hierarchies on MDX Cube Reports, page 124* for details.

Grouping related attributes: MDX cube source objects mapped to

MicroStrategy attribute forms are integrated into MicroStrategy as a string of characters. You can modify the data type used to map MDX cube data to attribute forms. This allows the MDX cube data to be correctly represented in MicroStrategy and facilitates the grouping of related attributes as the same attribute in a Report Services document. See *About Defining Column Data Types for MDX Cube Data, page 112* for details.

- SAP BW variables are mapped to MicroStrategy prompts. If an MDX cube contains key date variables, you must define them as key date variables to distinguish them from characteristic variables on date.

  For information on how SAP BW variables are converted into MicroStrategy prompts, see *Relating Objects from SAP BW to MicroStrategy, page 10*.

MicroStrategy objects can be managed like other objects

An MDX cube's equivalent structure and objects in MicroStrategy are represented in the **Logical View** right-hand column, with the standard MicroStrategy symbols for hierarchies, attributes, metrics, and so on.

For MicroStrategy objects, you can perform the following manipulations by right-clicking the object in the Logical view column and using the various options:

- **Edit** the attribute, metric, or prompt.

- **Rename** the attribute, metric, prompt, or hierarchy. If you want to map the object to an existing MicroStrategy object, you should use the Map feature rather than this Rename feature.

  During the import of MDX cubes, you can choose to have the MicroStrategy objects' names synchronized with their associated objects in the MDX cube source.

- **Map** the attribute, metric, or prompt to an existing attribute, metric, or prompt in the MicroStrategy project.
  - Attributes mapped to MDX cube data can be mapped to attributes in the MicroStrategy project that are part of the relational schema. See *Mapping MDX Cube Data to Project Attributes, page 106* for details.
- Metrics and prompts in MDX cubes can only be mapped to other managed object metrics and prompts that are mapped to MDX cube source data. See *Mapping SAP BW Variables to MicroStrategy Prompts, page 126* for details.
- Check the **Properties** of the attribute, metric, prompt, or hierarchy. The properties displayed when accessed from the Logical View column relate to MicroStrategy-specific properties such as the access control list, owner, and long description for the object.

Shared MDX Cube Objects

The data and objects that are shared in your MDX cube source are also shared as attributes in MicroStrategy projects. This maintains a consistent experience across all related MDX cubes imported into MicroStrategy.

For example, you import Cube1 and Cube2 into MicroStrategy. Both MDX cubes share data for year and category. In MicroStrategy, year data is mapped to an attribute named Year, and category data is mapped to an attribute named Category. If you change the name of the Year attribute in Cube1, this change is also reflected in Cube2. If you change the name of the Category attribute in Cube2, this change is also reflected in Cube1.

Attributes that are shared by multiple MDX cubes can have an effect on how and when attributes are imported for an MDX cube. For information on how these shared attributes can affect import behavior, see *Shared attribute effects on import behavior, page 89*.

Sharing metrics between MDX cubes

By default, each MDX cube creates its own metric objects to support the integration of the data into MicroStrategy. However, you can manually map metrics for multiple MDX cubes to the same metric so that they share a single metric. Sharing metrics between MDX cubes is required if you want to allow MDX cube reports to be able to switch the MDX cube that is used as the source of its data, as described in *Switching the MDX Cube for an MDX Cube Report, page 151*.

At least two MDX cubes must be imported into your MicroStrategy project. These MDX cubes must have data that can be mapped to the same metric.

To share metrics between MDX cubes

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time
2. In the **Physical View** column, choose **Measures** to view the metrics for the MDX cube.
3. Right-click a metric and choose **Map**.
4. Navigate to a different MDX cube that has similar metric data. Select the metric and click **Open**.
5. Once all relevant metrics are shared, from the MDX Cube Editor, click **Save and Close**.

Best Practices for Mapping MDX Cubes

When mapping MDX cubes to MicroStrategy objects:

- Rather than mapping all additional attribute forms one-by-one, you can have MicroStrategy automatically map all additional attribute forms for you. From the **View** menu, choose **Display All Columns** to display all the additional attribute forms. From the **Edit** menu choose **Map all attribute forms** to automatically map all additional attribute forms. You can modify these automatic mappings as required.

The Map all attribute forms option maps attribute data to managed object attributes. You must manually map all attribute forms if you are mapping your MDX cube data to project attributes.

- To display the entire MDX cube structure or only the top-level structure, from the **View** menu, you can use the **Expand All** or **Collapse All** options.
- To display or hide the SAP BW terms for each object, from the **View** menu, choose **Show Technical Names**. The Show Technical Names option applies to SAP BW MDX cubes only.
- After you have imported an MDX cube, you can perform the same mapping tasks available in the Cube Mapping tab of the MDX Cube Catalog by editing the MDX cube with the MDX Cube Editor. To use the MDX Cube Editor, right-click an MDX cube in Developer and choose **Edit**.
- For best practices to maintain object consistency between the two projects, see *Maintaining MDX Cubes Between Multiple Projects, page 96*.

Mapping MDX Cube Data to Project Attributes

After you have imported MDX cubes, you can use the automatically generated managed object attributes to define the levels of your MDX cube data. Alternatively, you can map MDX cube data to existing attributes in the MicroStrategy project that are part of the relational schema. Mapping MDX cube data in this way replaces the managed objects that are used to represent MDX cube data with attributes in the MicroStrategy project. Mapping MDX cube data to attributes in a MicroStrategy project that are part of a relational schema has the following benefits:

- Report designers can integrate the logical model of the project with the data in imported MDX cubes, thus creating a relation between the two sets of data. Data can then be joined across sources within a Report Services document. For example, if an MDX cube report and a standard report both use the same Year attribute, then Year can be used to group the data within a document.

- Administrators can search for dependents and manage access control lists (ACLs) for attributes that map both to the data warehouse and an MDX cube source.
- MicroStrategy security filters can be applied to attributes in MDX cube reports. For example, you can map an MDX cube level to the Year attribute in your project. If a user with a security filter on Year runs the MDX cube report that contains Year, the security filter on Year is applied.
- With the MicroStrategy MultiSource Option, MDX cube reports can be used to filter other standard reports in MicroStrategy. See *Using MDX Cube Reports to Filter Other Reports, page 180* for details.

Metrics and prompts mapped to MDX cube data cannot be mapped to objects that are part of a relational schema. These metrics and prompts can only be mapped to managed object metrics and prompts that are mapped to MDX cube source data. For example, three MDX cubes can share the same managed object metric named Revenue. However, none of these metrics can share the same object as a Revenue metric mapped to relational data in the project. Sharing metrics between MDX cubes is required if you want to allow MDX cube reports to be able to switch the MDX cube that is used as the source of its data, as described in *Switching the MDX Cube for an MDX Cube Report, page 151*.

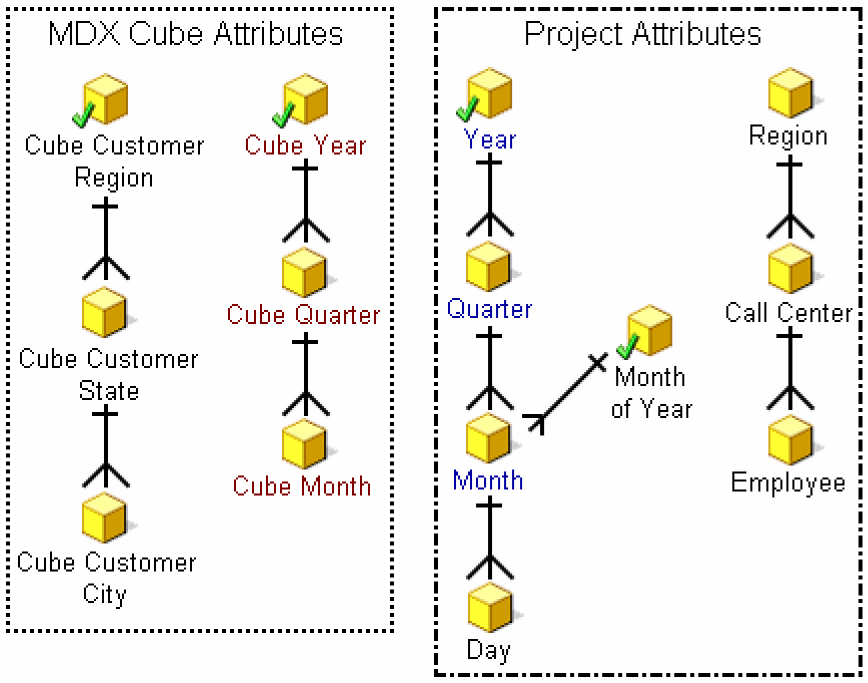
MDX cubes connected to SAP BW as an MDX cube source can contain variables. These variables are converted into prompts when imported into MicroStrategy (see *Relating Objects from SAP BW to MicroStrategy, page 10* for conversion information). If multiple MDX cubes contain the same variable, one MicroStrategy prompt can be mapped to more than one variable across MDX cubes. This enables a prompt to be displayed and answered only once when executing a Report Services document that uses these MDX cubes. For steps to map one prompt to variables in multiple MDX cubes, see *Mapping SAP BW Variables to MicroStrategy Prompts, page 126*.

Example 1: Unmapped MDX cube

You can map managed object attributes for your MDX cubes instead of using project attributes. This feature allows you to quickly start creating reports for your MDX cube data.

The drawback of an MDX cube mapped only to managed objects is that you cannot create a relation between your MDX cube data and other data in your project connected to a data source other than your MDX cube source. Since this relation is not created, you cannot join data from these different sources in a Report Services document and you cannot support project security filters in MDX cube reports.

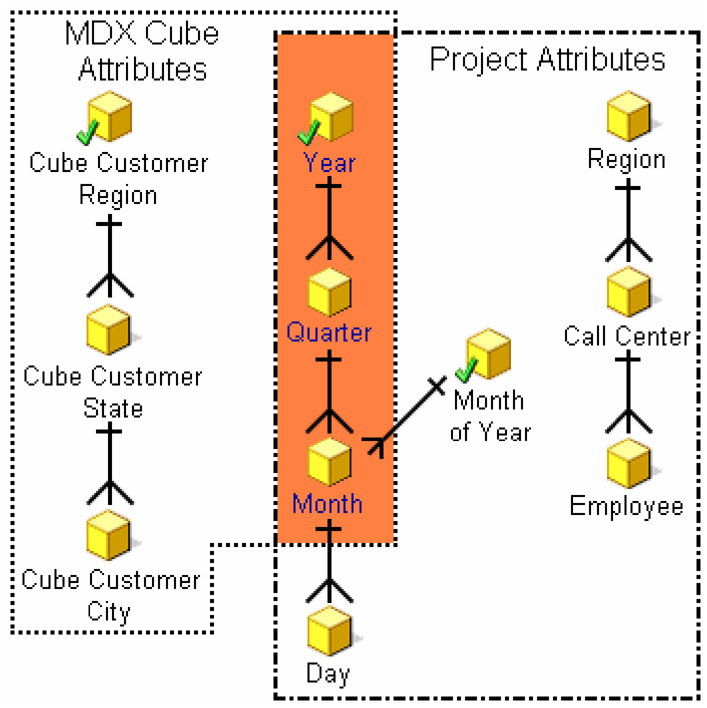
The diagram below shows two logical models. The one on the left exists in a specific MDX cube, and the one on the right exists in a MicroStrategy project. Although both models have a Time hierarchy, none of the individual attributes are shared.



Example 2: Partially mapped MDX cube

After an MDX cube source has been included in MicroStrategy as an MDX cube, you can map the attributes within the MDX cube to existing project attributes.

The example in the diagram below also shows two logical models. The difference between this example and the example above is that the MDX cube has been partially mapped so that it shares the attributes Year, Quarter, and Month. With this technique, you can create a Report Services document that contains Year, Quarter, and Month information for both your data warehouse and MDX cube source. In addition, any security filters for Year, Quarter, and Month are applied to MDX cube reports that include these mapped attributes.



The dimensions of MDX cubes are always shared. Therefore, when a level is mapped, that change applies to all the MDX cubes that share that dimension. In this case, changes to the Time dimension apply to MDX cubes in the project that contain this dimension.

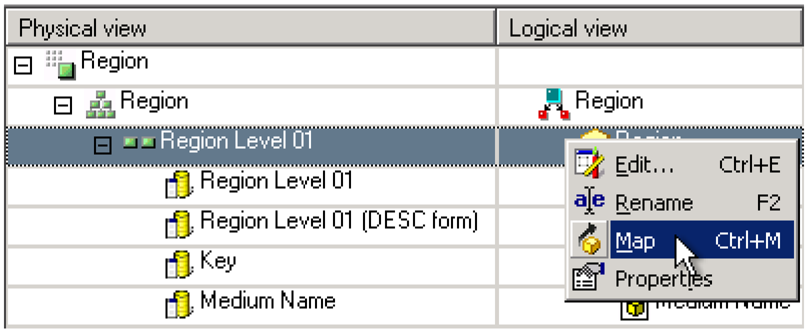
Best Practices for Mapping Cube Data to Project Attributes

✓ Import MDX cubes from your MDX cube source into MicroStrategy.

For general best practices of using the MDX Cube Catalog to map MDX cube data, see *Best Practices for Mapping MDX Cubes, page 105*.

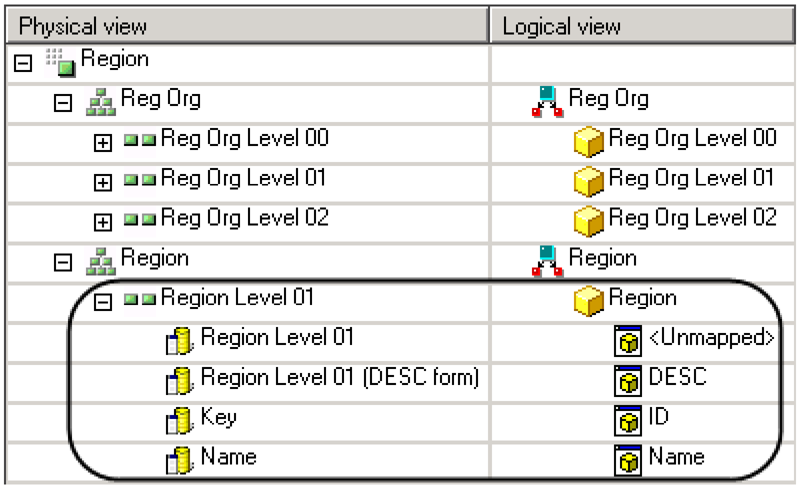
When mapping MDX cube data to project attributes, consider these recommendations:

- You can map MDX cube data to a project attribute by right-clicking the attribute in the Logical view column and choosing **Map**.



  You can then browse to the attribute within your relational project schema to map to the MDX cube data.

- When mapping SAP BW data to project attributes, you should map the SAP BW characteristics to project attributes rather than mapping the levels of SAP BW hierarchies. For example, in the scenario shown below you must map the Region project attribute to the level in the highlighted Region characteristic rather than to a level within the Reg Org hierarchy.



Physical view	Logical view
Region	
Reg Org	Reg Org
Reg Org Level 00	Reg Org Level 00
Reg Org Level 01	Reg Org Level 01
Reg Org Level 02	Reg Org Level 02
Region	Region
Region Level 01	Region
Region Level 01	<Unmapped>
Region Level 01 (DESC form)	DESC
Key	ID
Name	Name

The concept of SAP BW characteristics versus hierarchies is discussed in *Importing Levels and Suffixes for Characteristics, page 88*.

- To map MDX cube data to the ID form of a project attribute, you must adhere to the following guidelines:
    - The ID form of the project attribute must be mapped to the column in your MDX cube source you have created to relate the two systems of data. The columns must share the same data type. For example, the Key form in SAP BW can use the same numeric data type standards as is used most commonly for MicroStrategy attribute ID forms.

      The Key form is not displayed by default. Within the MDX Cube Catalog, from the **View** menu, choose **Display All Columns** to display all available forms for an MDX cube.

    - Once you map the correct column to the ID form of the project attribute, you must define the column data type of your MDX cube data as the same data type used for the attribute's ID form. This is because MDX returns all attribute data for MDX cube sources as strings by default. For information on and steps for defining column data types for MDX cube data, see *About Defining Column Data Types for MDX Cube Data, page 112*.

- You can map the columns to project attributes either when an MDX cube is first imported or at a later time. It is recommended that you immediately perform this mapping during the initial import to maintain a consistent reporting environment. This also prevents maintenance issues such as having to modify MDX cube reports when MDX cubes are modified after the reports are created.

  If you map a column to the incorrect project attribute, do one of the following:

  - Once you save your changes, you can unmap the column from the project attribute.
  - You can close the MDX Cube Catalog without saving your changes. The mapping is not changed for the column. Be aware that any other changes made after the last time you saved your changes will also be lost.

- If you need to unmap a column that was previously mapped to a project attribute, it is recommended to unmap the column and then save immediately after performing this unmapping. This ensures that the unmapping is processed correctly. You can then open the MDX cube and perform any additional mapping or unmapping of columns as needed.

About Defining Column Data Types for MDX Cube Data

You can define the column data type that is applied to a column of MDX cube data mapped to an attribute. This allows the MDX cube data to be correctly represented in MicroStrategy and facilitates the following:

- Group related attributes, from MDX cubes as well the data warehouse, as the same attribute in a Report Services document. This is discussed in more detail in this section.
- Use value prompts to qualify on your MDX cube data. If you plan to use a value prompt (date, numeric, text, or big decimal) to qualify on an attribute form imported from an MDX cube source, you must define the attribute

form with a data type that matches the value prompt. For example, an Integer column data type can be qualified on using a numeric value prompt. See *Define Column Data Types for MDX Cube Data* for details.

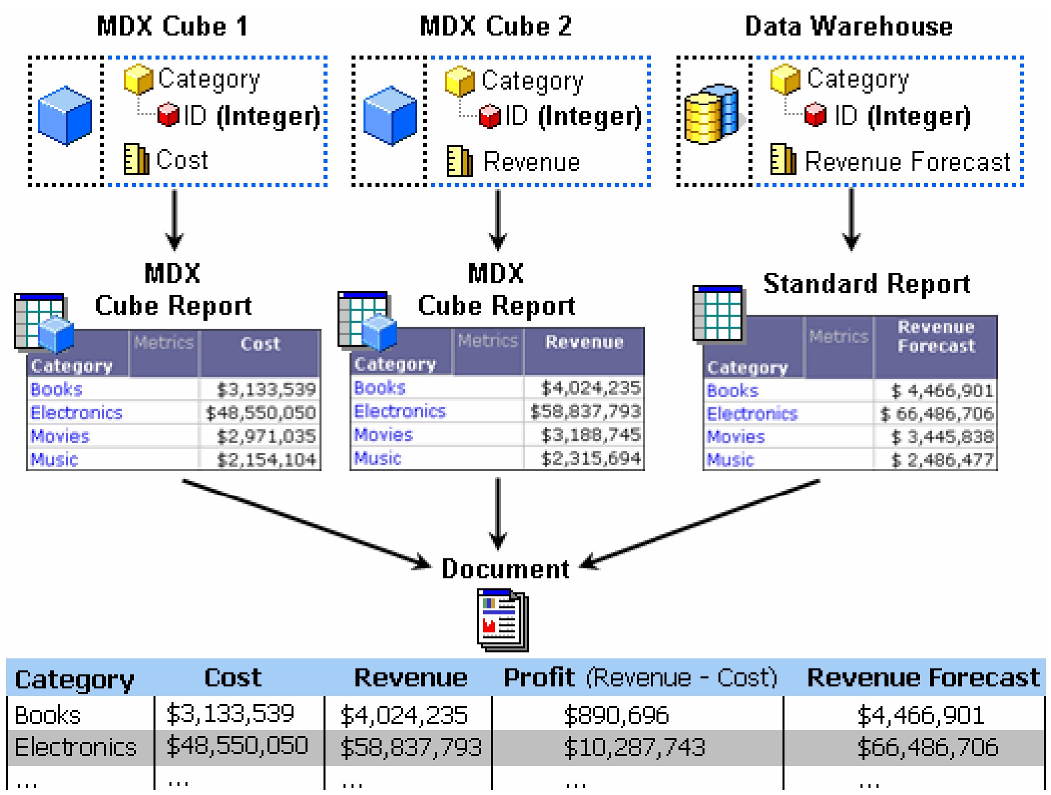
- Support date data from your MDX cube source in MicroStrategy. This enables you to filter and qualify on your MDX cube source date data using static and dynamic date qualifications. See *Supporting MDX Cube Source Date Data in MicroStrategy, page 120* for details.

When MDX cube data is mapped to MicroStrategy objects, MicroStrategy retrieves the column data type through MDX. In the case of MDX cube data that is mapped to attributes, the columns are often returned as a string of characters. This can be the case even with ID columns of data that are commonly of a numeric type such as integer.

The column of data that is automatically mapped to attribute ID columns in MDX cubes is returned as a character string. The data type of this column cannot be defined to anything other than the default data type because it is not well suited for other data types. If you want to map a column with a numeric or other data type to the ID form of an attribute, you should use a different form such as the key form.

MDX cube data that is mapped to MicroStrategy metrics is automatically converted to a numeric data type and thus does not need its column data type to be manually set.

For example, you have two MDX cubes that map data to a Category attribute in MicroStrategy. The ID attribute form for Category is returned as a string by default. However, you know that its associated MDX cube column is of type integer and set the data type accordingly in each MDX cube. You can then create MDX cube reports for these MDX cubes. By setting the Category ID attribute form to read the MDX cube data as an integer, you can then include the two MDX cube reports as datasets of a Report Services document and group the Category attribute data. You can also add a standard report, drawing data from a data warehouse, as a dataset of the document to combine its data on the same display.



Category	Metrics Cost
Books	$3,133,539
Electronics	$48,550,050
Movies	$2,971,035
Music	$2,154,104

Category	Metrics Revenue
Books	$4,024,235
Electronics	$58,837,793
Movies	$3,188,745
Music	$2,315,694

Category	Metrics Revenue Forecast
Books	$ 4,466,901
Electronics	$ 66,486,706
Movies	$ 3,445,838
Music	$ 2,486,477

Category	Cost	Revenue	Profit (Revenue - Cost)	Revenue Forecast
Books	$3,133,539	$4,024,235	$890,696	$4,466,901
Electronics	$48,550,050	$58,837,793	$10,287,743	$66,486,706
...	...	...	...	...

Notice that the Category ID form is defined as the same data type (Integer) in each data source. Without this setup, the Category data from each data source cannot be displayed on a single document.

In addition to displaying data from different data sources on the same document, defining data types lets you perform calculations on metrics from different data sources. In the document shown above, the Profit metric is calculated by creating a calculated expression in the document that subtracts the Cost metric of MDX Cube 1 from the Revenue metric of MDX Cube 2.

The image shows a scenario of displaying both MDX cube data and data from a data warehouse on one document. If you only include MDX cube data on a document, you can use the default attributes created by MicroStrategy when importing your MDX cubes. However, to include MDX cube data and data from the data warehouse together on a document, you must map your MDX cube data to attributes that are part of the project's relational schema. See *Mapping MDX Cube Data to Project Attributes, page 106* for details.

For more information about defining and modifying the column types, see:

After defining the column data types, you can create MDX cube reports that can be displayed on a document with other MDX cube reports and standard reports. For information on how to create a document that displays data from multiple data sources, and information on creating calculated expressions see the Document Creation Help.

Define Column Data Types for MDX Cube Data

To Define Column DataTypes for MDX Cube Data

You can define column data types for MDX cube data when you are mapping MDX cube data to MicroStrategy objects. The following procedure assumes you are defining column data types as part of the mapping procedure for an MDX cube using the MDX Cube Catalog. However, you can define column data types as a later modification using the MDX Cube Editor.

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.
2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.
3. Choose **Cube Mapping**.
4. From the **Catalog\Cube** drop-down, choose an MDX cube to map to MicroStrategy objects.
5. In the **Physical view** column, expand the MDX cube data until you find the MDX cube column data for which to manually set the data type.

6. If you want to include MDX cube data and data from the data warehouse together on a document, you must map your MDX cube data to attributes that are part of the project's relational schema. See *Mapping MDX Cube Data to Project Attributes, page 106* for details.

   Using the automatically created managed object attributes enables you to combine data from multiple MDX cubes on the same document, but no data from the data warehouse can be included.

7. Under **Logical view**, right-click the MicroStrategy object mapped to the MDX cube column data and choose **Data Type**.

8. Clear the **Use default from source** check box.

9. From the **Datatype** drop-down, choose which data type to map the MDX cube data as in MicroStrategy.

   The Integer data type is commonly used as an ID form data type.

10. Depending on the data type chosen, specify the byte length, precision, and scale for the data type.

11. Click **OK**.

Resolving Incompatible Data Type Errors

Defining your MDX cube data with an incompatible data type can cause errors to occur when running a MicroStrategy MDX cube report. An MDX cube report that includes MDX cube data mapped to an incompatible data type fails and no data is returned. When an MDX cube report fails for this reason, an error message is displayed that identifies the data that has been mapped to an incompatible data type.

Modify the Attribute Form Format Type

To modify the attribute form format type

1. In the MDX Cube Catalog, right-click the attribute mapped to the column you defined as a date data type, and select **Edit**.
2. Choose **Forms** > **Attribute forms**, select the form you defined as a date data type, and click **Modify**.
3. Choose **Form format** > **Type** > **Date**.
4. Click **OK**.
5. If an inconsistent data type warning message is displayed, click **Yes**.
6. Click **Save and Close** twice.

Preserving Attribute Element Orders from MDX Cube Sources

When data is integrated from your MDX cube source to MicroStrategy, the order of data mapped to attribute elements in MicroStrategy does not always reflect the same order of the data in your MDX cube source.

Preserving the order of data as it exists in your MDX cube source can be helpful for various reasons. For example, this can help support financial balance sheet analysis in which you always want to see your accounts receivable information above your accounts payable information.

To preserve the order of data as it exists in your MDX cube source, you must define dimensions within MDX cubes to return the order of data when integrated into MicroStrategy.

Dimensions in MDX cube sources are not directly mapped to MicroStrategy objects. However, the levels within a dimension in an MDX cube source are mapped to hierarchies and attributes in MicroStrategy.

By default, the order of data is not integrated into MicroStrategy. This is because the need to preserve the order of data is commonly unnecessary and only used on a case-by-case basis, and integrating this order information requires metadata space.

Before you can preserve the order of MDX cube source data in MicroStrategy, you need to import MDX cubes from your MDX cube source into MicroStrategy. For steps to import MDX cubes, see *Importing MDX Cubes, page 83*.

To preserve and retrieve attribute element orders for MDX cube sources

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.

2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.

3. Choose **Cube Mapping** > **Catalog\Cube** and then choose the MDX cube to integrate the order of data for into MicroStrategy.

4. You must save the MDX cube before you can modify the properties of a dimension. Choose **File** > **Save**.

5. In the **Physical view** column, right-click a dimension to integrate the order for all data within the dimension into MicroStrategy, and then choose **Properties**.

6. On the **Hierarchies** tab, choose the **The order of the hierarchy nodes should be preserved from the source** check box. Choosing this check box marks this dimension to have the order of its data integrated into MicroStrategy.

7. To retrieve the order of data from the MDX cube source, click **Retrieve hierarchy structure**. If the order of data in your MDX cube source

changes, you can update the order for all dimensions in an MDX cube that are defined to have the order of data integrated into MicroStrategy.

8. Click **OK**.

9. Click **Save and Close**.

The order of data is now integrated into MicroStrategy. However, MDX cube reports are not sorted using this order by default. A report designer must define an MDX cube report to sort using the order integrated from the MDX cube source, as described in *Sorting on Attribute Element Orders from MDX Cube Sources, page 176*.

Updating the order of attribute elements

You can choose to preserve the order of data that is mapped to attribute elements in MicroStrategy for various MDX cubes. Once you have done so, you can update these orders for MDX cubes in MicroStrategy when they are changed in your MDX cube source.

✓ The order of data in an MDX cube source is only updated for dimensions that are defined to integrate the order of data into MicroStrategy.

To update the order of attribute elements

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.

2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.

3. On the **Cube Selection** tab, in the **Selected Cubes** area, browse to and right-click an MDX cube to update the order of its data in MicroStrategy, and then choose **Update hierarchy structure**. The order of data is updated in MicroStrategy for all dimensions that are defined to integrate the order of data into MicroStrategy.

4. Click **Save and Close**.

Supporting MDX Cube Source Date Data in MicroStrategy

You can maintain a consistent date format when mapping date data you have stored in your MDX cube source into MicroStrategy. This enables you to view the data as its intended date data type, as well as filter and qualify on date data using static and dynamic date qualifications.

Date data represents a given day using various formats that include the month, day, and year of a given day.

The following procedures explain how to support MDX cube source date data in MicroStrategy:

After these steps are completed, you can begin to create MDX cube reports that include the date data defined for the attribute form mapped to the MDX cube source date data. For information on how date data can be filtered on using static and dynamic date filters, see *Filters on MDX Cube Reports, page 157*.

Support MDX Cube Source Date Data in MicroStrategy

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.

2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.

3. Choose **Cube Mapping**.
4. From the **Catalog\Cube** drop-down list, select the MDX cube to define as a date data type.
5. In the **Physical view** column, expand the MDX cube data until you find the MDX cube column data for which to define the data type.

   ⚠ The column of data that is automatically mapped to attribute ID columns in MDX cubes is returned as a character string. The data type of this column cannot be defined to the Date data type.

6. From the **Logical view** column, right-click the MicroStrategy object mapped to the MDX cube column data and select **Data Type**.
7. Clear the **Use default from source** check box.
8. Choose **Datatype** > **Date**.
9. Click **OK**.

Modify the Attribute Form Format Type

To modify the attribute form format type

1. In the MDX Cube Catalog, right-click the attribute mapped to the column you defined as a date data type, and select **Edit**.
2. Choose **Forms** > **Attribute forms**, select the form you defined as a date data type, and click **Modify**.
3. Choose **Form format** > **Type** > **Date**.
4. Click **OK**.
5. If an inconsistent data type warning message is displayed, click **Yes**.
6. Click **Save and Close** twice.

Modify the MicroStrategy VLDB Property

To modify the MicroStrategy VLDB property

1. Right-click the project that contains your MDX cubes and select **Project Configuration**.
2. Choose **Project definition** > **Advanced**.
3. Choose **Project-Level VLDB settings** > **Configure**.
4. Choose **MDX** > **Format for date/time values coming from data source**.

   The default date format for MDX cube sources is displayed on the right. If this date format does not match the date formats used in your MDX cube source, clear the Use default inherited value check box. You can then enter the date format used in your MDX cube source.

   You can also define the date formats used in an individual MDX cube report. If you define this VLDB property for an MDX cube report, the definition for the MDX cube report takes precedence over the definition for the project. However, the definition of the VLDB property for the project is applied to all MDX cube reports that select to use the default inherited date format.
5. Click **Save and Close**.
6. Click **OK**.

Defining Unbalanced and Ragged Hierarchies

By default, all hierarchies of an MDX cube are treated as balanced hierarchies. However, if you know that the structure of a hierarchy is unbalanced or ragged, you must set the hierarchy's properties to reflect its structure.

The different characteristics of hierarchical sets of data include:

- **Balanced** hierarchies have an equal number of levels in each branch of the hierarchy. For example, in a Product hierarchy that includes Category, Subcategory, and Item, each branch descends to a particular item, which is at the lowest level.
- **Unbalanced** hierarchies have at least one branch that does not descend to the lowest level. For example, in a Time hierarchy that includes Year, Quarter, and Month, one branch might only have data down to the Quarter level.
- **Ragged** hierarchies have at least one branch that includes a member whose logical parent is not the level immediately above that member. For example, a Product hierarchy may contain the levels Category, Subcategory, and Item, but Item number 22 does not have a Subcategory associated with it. When Category, Subcategory, and Item are displayed on the report, there is an empty cell for the Subcategory of Item number 22.
- **Unbalanced and ragged** hierarchies include at least one branch that does not descend to the lowest level and one branch that includes a skipped level.

To prevent inaccurate results when applying certain types of filters, an unbalanced or ragged hierarchy must be defined.

Before you can define a hierarchy in an MDX cube as unbalanced or ragged, you need to import MDX cubes from your MDX cube source into MicroStrategy. For steps to import MDX cubes, see *Importing MDX Cubes, page 83*.

To define a hierarchy as unbalanced or ragged

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.
2. Choose **Schema** > **MDX Cube Catalog**.

If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.

3. Choose **Cube Mapping**.
4. From the **Catalog\Cube** drop-down, choose the MDX cube that contains the unbalanced or ragged hierarchies.
5. In the MDX Cube Catalog, right-click the hierarchy name in the Physical View column and choose **Properties**.

   A hierarchy in the Physical View column is represented with a green stacked boxes symbol ( ).

6. Choose **Hierarchies** and select the check box **This hierarchy is unbalanced or ragged**.
7. Click **OK**.
8. Click **Save and Close**.

Displaying Hierarchies on MDX Cube Reports

Report designers can include hierarchies within an MDX cube on the templates of MDX cube reports. At report runtime, any hierarchies included on an MDX cube report display attributes that are a part of that hierarchy.

When an MDX cube report includes a hierarchy, the attributes that are displayed for that hierarchy are determined by two factors:

- The default number of attributes to display for a hierarchy.

  You can define this default for a hierarchy when mapping data from your MDX cube source to an MDX cube in MicroStrategy. When you define the number of attributes to display for a hierarchy, you define the number of attributes to display from the highest level sequentially down to the lowest level. This is because attributes for a hierarchy are displayed in order from the highest level (first) down to the lowest level (last).

For example, you have a Geography hierarchy with Country, Region, and City attribute levels. You define the hierarchy to display only one attribute on reports. When a report with the Geography hierarchy on the template is executed, only Country is displayed because it is the highest level attribute in the hierarchy. Defining Geography to display two attributes displays Country and Region, while defining Geography to display three attributes displays Country, Region, and City.

- The attribute level defined in the report filter of the report. A report designer may include one or more attributes in a report filter that are at a lower level than is set for a hierarchy. For more information on how the report filter affects the attributes that are displayed for a hierarchy on a report, see *Hierarchies on MDX Cube Reports, page 152*.

The lower attribute level of the two factors is used as the attribute level displayed on the report. When mapping and configuring MDX cubes, you should set the default number of attributes to display for a hierarchy so that it fits the majority of report requirements. For more information on how this default interacts with report filters to determine the attributes displayed on a report, see *Hierarchies on MDX Cube Reports, page 152*.

Only the default report displays forms for each attribute are shown when attributes are displayed as part of a hierarchy on a report. To modify the attribute forms that are displayed for each attribute when displayed as part of a hierarchy, you must modify the report display forms for the attributes using the Attribute Editor. For steps to modify an attributes report display forms, see the Project Design Help.

Before you can define the default number of attributes to display on reports for a hierarchy, you need to import MDX cubes from your MDX cube source into MicroStrategy. For steps to import MDX cubes, see *Importing MDX Cubes, page 83*.

To define the default number of attributes to display on reports for a hierarchy

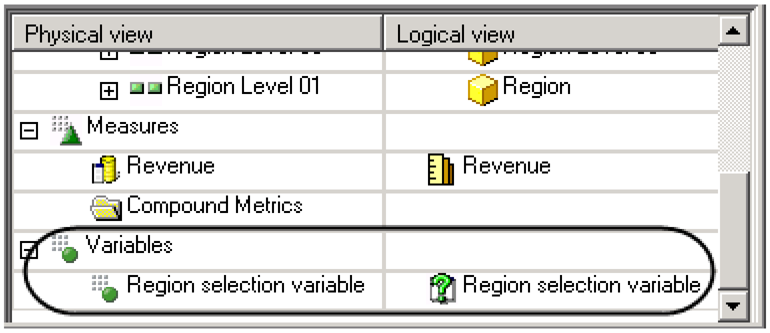
1. In Developer, log in to a project connected to your MDX cube source.
2. Navigate to the cube source's **Data Explorer** and browse to an MDX cube.
3. Right-click the MDX cube and choose **Edit**.
4. In the **Physical view** column, right-click a hierarchy and choose **Properties**.
5. Choose **Hierarchy**.
6. Define the **Set default hierarchy display depth** to the depth of attributes to display for the hierarchy on reports.

   A value of 0 displays only the highest level attribute for the hierarchy, while a value of 1 displays the highest level attribute and the next highest level attribute for the hierarchy, and so on.
7. Click **OK**.
8. Click **Save and Close**.

Mapping SAP BW Variables to MicroStrategy Prompts

Variables in SAP BW allow users to enter values as parameters for the queries on a cube. SAP BW variables are represented as MicroStrategy prompts when they are imported into the MicroStrategy environment. For information on how SAP BW variables are converted into MicroStrategy prompts, see *Relating Objects from SAP BW to MicroStrategy, page 10*.

The mapping between variables and prompts can be viewed in the Cube Mapping tab of the MDX Cube Catalog or in the MDX Cube Editor. Both interfaces list all the variables that were converted to prompts.



To allow attribute element searches on MDX cube reports for SAP BW cubes that include variables, within your SAP BW system, you must either define the variables as optional, or provide a default answer for the required variables. The MicroStrategy prompt that is mapped to the SAP BW variable can be either optional or required.

See the following topics to learn how to define SAP BW variable as key date variables, support variable qualifications, and to use one prompt for variables in separate cubes:

You can also perform the following mapping and configuration tasks for SAP BW variables and their associated MicroStrategy prompts:

- To view any variable's properties, right-click its name in the Physical View column, and then choose **Properties**.
- To edit prompts created by the import process, right-click the prompt in the Logical View column, and choose **Edit**.
- To rename a prompt, right-click the prompt in the **Logical View** column and select **Rename**.
- To map an SAP BW variable to a specific prompt in the MicroStrategy project, right-click the project and select **Map**.

Supporting SAP BW Key Date Variables

If you use any SAP BW key date variables in your query, you need to manually set the variables as key date variables. This must be done to distinguish them from simple characteristic variables for dates.

✓ Import MDX cubes from your MDX cube source into MicroStrategy.

To define SAP BW variables as key date variables

1. In Developer, log in to a project connected to an MDX cube source.
2. Right-click an MDX cube with a key date variable and choose **Edit**.

   If the Read Only dialog is displayed, choose **Edit** and click **OK**.

   If you are only given the option of opening the MDX Cube Editor in read only mode, this means another user is modifying the project's schema. You cannot open the MDX Cube Editor in edit mode until the other user is finished with their changes and the schema is unlocked.

   For information on how you can use read only mode and edit mode for various schema editors, see the Project Design Help.
3. In the **Physical View** column, right-click the variable and choose **Properties**.
4. On the **Variable** tab, select the **Set Key Date** check box, and then click **OK**.
5. Click **Save and Close**.

Supporting SAP BW Variable Qualifications

If you use any SAP BW variables with expression qualifications in your query, you can define which form is used to evaluate the variable's qualification. This gives you the flexibility to use either the key form or the ID form from your SAP BW data source to evaluate the variable's qualification.

✓ Import MDX cubes from your MDX cube source into MicroStrategy.

To define the form used to evaluate an SAP BW variable's qualification

1. In Developer, log in to a project connected to an MDX cube source.
2. Right-click an MDX cube with a variable and select **Edit**. If the Read Only dialog box is displayed, select **Edit** and click **OK**.

   If you are only given the option of opening the MDX Cube Editor in read only mode, this means another user is modifying the project's schema. You cannot open the MDX Cube Editor in edit mode until the other user is finished with their changes and the schema is unlocked.

   For information on how you can use read only mode and edit mode for various schema editors, see the Project Design Help.
3. In the **Physical view** column, expand the MDX cube data until you find the MDX cube column data for which to define as the form to evaluate the SAP BW variable's qualification.

   You can define either the key form or the ID form from your SAP BW data source as the form used to evaluate the variable's qualification.
4. From the **Logical view** column, right-click the MicroStrategy attribute form mapped to the MDX cube column data and choose **Variable Qualification Form**.
5. Click **Save and Close**.

Using One Prompt in Documents for Variables in Separate MDX Cubes

Using one prompt in documents for variables in separate MDX cubes

One MicroStrategy prompt can be mapped to multiple SAP BW variables in different MDX cubes. This enables a prompt to be displayed and answered only once when executing a Report Services document using these MDX cubes. If you do not map the same prompt to the variables, the user must answer the prompt for each variable.

You should be aware of the following when mapping variables from different MDX cubes to the same prompt:

- When you map a variable to a prompt, the prompt that you select replaces the prompt that was previously mapped to the variable.
- Variables mapped to the same prompt do not have to be identical. However, the variables must be similar enough that the prompt can complete any variables it is mapped to with the same prompt answer.
- Prompts are automatically mapped to a variable in an MDX cube. You can then map these automatically created prompts to variables in other MDX cubes. This ensures that the prompt is defined to work correctly with an SAP BW variable.

  For example, you import MDX cubes A, B, and C, which all have a variable that you want to map to the same prompt. The prompt that you want to use is mapped to the variable in MDX cube A. You can edit MDX cube B and MDX cube C to map the variables in each of these MDX cubes to the prompt in MDX cube A. All of the variables in the three MDX cubes are then mapped to the same prompt.

  ✓ Import MDX cubes from your MDX cube source into MicroStrategy.

To map multiple variables to one prompt

1. In Developer, log in to a project connected to an MDX cube source.
2. Right-click an MDX cube that contains a variable and select **Edit**. If the Read Only dialog box is displayed, select **Edit** and click **OK**.

   If you are only given the option of opening the MDX Cube Editor in read only mode, this means another user is modifying the project's schema. You cannot open the MDX Cube Editor in edit mode until the other user is finished with their changes and the schema is unlocked.

For information on how you can use read only mode and edit mode for various schema editors, see the Project Design Help.

3. In the **Logical View** column, right-click the prompt mapped to the variable and select **Map**.
4. Browse to and select the prompt to map to the variable and click **Open**. The prompt you select replaces the prompt previously mapped to the variable.
5. Click **Save and Close**.

Creating Metrics from MDX Cube Data

When you map your MDX cube data into MicroStrategy, you can take advantage of MDX (MultiDimensional eXpressions) to create metrics. Metrics created with MDX combine the robust set of MDX functions and expressions with MicroStrategy analytical tools such as prompts. You can also use basic arithmetic expressions to create these metrics from MDX cube data.

Once you create metrics using these techniques you can include them in your MicroStrategy reports and report filters in the same ways that you can use any MicroStrategy metric. You can also use prompts in these compound and custom MDX metrics (see *Using Prompts within MDX Cube Metrics, page 137*). The metrics created in this way for an MDX cube are stored in a Compound Metrics folder within the Metrics folder for the MDX cube.

Metrics created to map to your MDX cube data are related only to their associated MDX cube. Therefore, these metrics cannot be directly integrated with data from a separate relational data source, except by using calculated expressions in Report Services documents. For information on creating calculated expressions, see the *Designing and Creating Documents* section of the Document Creation Help.

You can create metrics that map to MDX cube data using either of the following techniques:

- **Compound metrics**: A compound metric is any MicroStrategy metric with an expression that includes a MicroStrategy metric and an arithmetic expression. The expression can be as simple as a metric multiplied by a constant value, such as `Discount * 1.5`, where `Discount` is a metric mapped to data in the MDX cube. These metrics can also reference multiple MicroStrategy metrics within the MDX cube with an expression such as `Revenue - Total Expenses`, where Revenue and Total Expenses are both metrics, to build a Profit metric.

  You can use MicroStrategy analytical and aggregate functions such as `SUM`, `COUNT`, and `AVG` with metrics mapped to MDX cube data only if the metric you create is defined as a smart metric. If you do not make the metric a smart metric you can only use basic operators (`+`, `-`, `/`, `*`, and so on). For general information on smart metrics, see the Basic Reporting Help. For examples of smart metrics, see the Advanced Reporting Help.

- **MDX customization**: Rather than relying only on MicroStrategy to create MDX to return data from your MDX cube source, you can create your own custom MDX to return data for a metric. This technique allows you to use MDX functions and flexibility to query and report on your MDX cube data. The MDX you create is passed to your MDX cube source to be executed and to return the data. You can reference one or more MicroStrategy metrics mapped to MDX cube data using custom MDX just as you can with a standard arithmetic expression. To use MDX to create your calculated measures you must enclose MDX in double quotes (`""`). For tips and insights on how to build analysis with MDX in MicroStrategy, see *How to Build Analysis into Metrics with Custom MDX, page 134*.

You can create these metrics during the initial importing and mapping procedure of your MDX cube data with the MDX Cube Catalog. These metrics can also be created as a later modification to an MDX cube with the MDX Cube Editor.

The following procedure uses the MDX Cube Catalog:

Create a Metric from MDX Cube Data with MDX and Compound Metric Techniques

Import MDX cubes from your MDX cube source into MicroStrategy

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.
2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.
3. Choose the **Cube Mapping** tab.
4. From the **Catalog\Cube** drop-down, choose the MDX cube.
5. From the **Edit** menu, choose **Add New Compound Metric**.
6. Create the expression for your metric:
   - If you are creating a compound metric, you can drag and drop metrics from the MDX cube's Metrics folder. This includes any required constants, arithmetic operators, and MicroStrategy analytical and aggregate functions in your new metric expression. For example, if you have Revenue and Cost metrics in your MDX cube you can create the expression `Revenue - Cost` to create a Profit metric.

     You can use MicroStrategy analytical and aggregate functions with metrics mapped to MDX cube data only if the metric you create is defined as a smart metric. See the Advanced Reporting Help for information on enabling smart metrics.

- If you are creating a metric using custom MDX, enter your custom MDX in the **Definition pane** of the Metric Editor. Make sure to enclose the entire expression in double quotes. For example, you can enter the following:

  ```
  "[Measures].[Discount Amount] * 1.5"
  ```

  You cannot validate MDX in the Metric Editor as you can for a standard expression that is not enclosed by double quotes. Validating MDX verifies that the entire expression is enclosed in double quotes; it does not validate the syntax of the expression.

  For an example of creating a metric that includes a prompt, see *Using Prompts within MDX Cube Metrics, page 137*.

7. Click **Save and Close**.
8. In the **Object name** text field, enter a name for your metric.
9. Click **Save** .
10. Click **Save and Close**.

How to Build Analysis into Metrics with Custom MDX

You can build sophisticated analysis into your MDX cube metrics by creating your own custom MDX. This allows you to further combine the analysis capabilities of MDX and MicroStrategy. This section provides some tips and best practices on how to build analysis into metrics with custom MDX.

Creating such analysis requires appropriate knowledge of both MDX and MicroStrategy. MicroStrategy does not validate any custom MDX created by users to build metrics for MDX cubes.

MDX has strict rules about the inclusion or exclusion of dimensions, hierarchies, and attributes on the report template and in custom MDX formulas. Some MDX formulas expect related attributes to exist on the template, and they may return incorrect results (or an error) if the attributes are omitted. Other formulas may return unexpected results if the attributes are included. As a result, certain custom MDX formulas may not be appropriate for ad-hoc reporting and you should be aware of the possible limitations of the custom MDX you create.

Creating your own custom MDX allows you to draw further analysis from your MDX cube source into MicroStrategy. Any expressions that are valid in a `WITH MEMBER` clause may be used, allowing metrics built in MicroStrategy to employ the data manipulation capabilities of the MDX cube source. The custom MDX is placed into a `WITH MEMBER` clause defining a member of the Measures collection for the scope of the query.

To use MDX to create your metrics, you must enclose MDX in double quotes (""). For example, "`[Measures].[Total Sales]`" is valid syntax for a metric defined with MDX, returning the Total Sales data from an MDX cube.

Since MDX is passed to and run against your MDX cube source, you must use the names and identifiers used in the MDX cube source to identify the data to be retrieved. For SAP BW, the technical name should be used.

In the example, "`[Measures].[Total Sales]`" in your MDX cube source, your metric data is identified as Total Sales and it mapped to a metric named Revenue in MicroStrategy. When you are creating custom MDX to retrieve this data from your MDX cube source, you must use the identifier for the data within the MDX cube source.

You can also perform basic arithmetic in your MDX, such as applying a multiplier to the Total Sales data: "`[Measures].[Total Sales] * .06`"

Along with these simple expressions, you can also use MDX functions to create more advanced analysis. When you include an MDX function in your custom MDX, the function is passed to the MDX cube source and processed as a pass-through function. For example, you can use the MDX year-to-date (YTD) function to create transformation-style analysis on your MDX cube data: "`sum(YTD([Quarter].CurrentMember), [Measures].[Profit])`". This expression returns year-to-date values by quarter for profit data.

See the following topics about specific methods for building analysis into your metrics:

Creating Transformation-Style Analysis

You can create transformation-style analysis with custom MDX by using two types of MDX functions. MDX provides functions:

- `PrevMember` and `NextMember` to move to the previous or next element in the same attribute.
- Such as `MTD`, `WTD`, or `YTD` (month, week, and year to date) to aggregate data based on time hierarchy models.

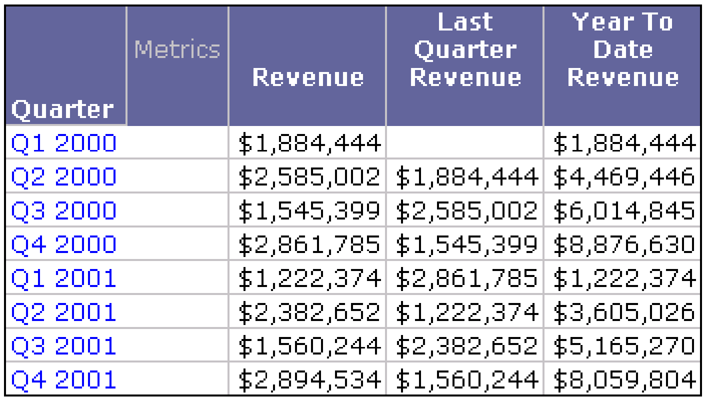
The following custom MDX examples include a one-to-one transformation to display the previous quarter's revenue, and a many-to-one transformation calculating year-to-date revenue:

- `"([Measures].[REVENUE], [Quarter].CurrentMember.Prevmember)"`

  Displays the revenue data for the previous quarter. The Last Quarter Revenue metric shown in the report below is mapped to this custom MDX formula.

- `"sum(YTD([Quarter].CurrentMember), [Measures].[REVENUE])"`

  Displays the total revenue data for all quarters of a given year up to and including the current quarter. The Year To Date Revenue metric shown in the report below is mapped to this custom MDX formula.



Metrics	Revenue	Last Quarter Revenue	Year To Date Revenue
Quarter			
Q1 2000	$1,884,444		$1,884,444
Q2 2000	$2,585,002	$1,884,444	$4,469,446
Q3 2000	$1,545,399	$2,585,002	$6,014,845
Q4 2000	$2,861,785	$1,545,399	$8,876,630
Q1 2001	$1,222,374	$2,861,785	$1,222,374
Q2 2001	$2,382,652	$1,222,374	$3,605,026
Q3 2001	$1,560,244	$2,382,652	$5,165,270
Q4 2001	$2,894,534	$1,560,244	$8,059,804

Using Prompts and ApplySimple Statements

Using the ApplySimple function, you can include prompts in your MDX to provide dynamic analysis on your MDX cube data. For basic information and examples of the ApplySimple function, see the Functions Reference.

The example below shows the basic structure of an ApplySimple statement to create metrics with custom MDX:

```
ApplySimple("MDX expression with placeholders for
objects",object0,object1,...,objectN)
```

A simple application of this technique is to use a constant value prompt in your project as a multiplier of metric data:

```
ApplySimple("([Measures].[Total Sales] / #0)" ,?valueprompt)
```

In the example syntax above, `#0` is a placeholder in the MDX expression for the value prompt. The syntax for including a prompt as an object to replace a placeholder is `?promptname`.

You can also use this technique with the conditional metrics techniques. For example, rather than always returning the revenue data for electronics, you can allow users to choose what category to view revenue for. To provide this analysis to users, you can include an element list prompt on the Category attribute of the MDX cube:

```
ApplySimple("([Measures].[Revenue],#0)", ?elementlistprompt)
```

Using Prompts within MDX Cube Metrics

If you are creating new metrics in your MDX cube, you can also include MicroStrategy prompts with the metrics. When the metrics are included on a report and the report is run, the prompts are displayed to the user for completion. This adds flexibility to your queries, allowing users to determine the data to see on the report.

This section covers the general requirements and process to include prompts in MDX cube metrics. See *How to Build Analysis into Metrics with*

Custom MDX, page 134 for information about how to include prompts within custom MDX of MDX cube metrics.

For metrics created with compound metric techniques without any custom MDX, you can include a prompt in the metric definition. For metrics created using MDX expressions, you must use the ApplySimple function to include prompts in the metric definition.

The following types of prompts can be included with MDX cube metrics:

- Element list prompts defined on an attribute of the associated MDX cube
- Value prompts
- Object prompts defined on objects of the associated MDX cube

For information on how prompts can be included in an MDX cube report and the supported prompt types, see *Prompts on MDX Cube Reports, page 165*. For a review of all prompt types, see the Basic Reporting Help.

Before you can use prompts in MDX cube metrics, you need to import MDX cubes from your MDX cube source into MicroStrategy. For steps to import MDX cubes, see *Importing MDX Cubes, page 83*.

To use prompts in MDX cube metrics

1. In Developer, log in to a project that is connected to an MDX cube source. Only one user can edit a project at a time.
2. Choose **Schema** > **MDX Cube Catalog**.

   If the project connects to more than one MDX cube source, choose an MDX cube source database instance from the **Select the Database Instance** drop-down list, and then select an MDX cube source database instance and click **OK** to open the MDX Cube Catalog.

3. Choose **Cube Mapping**.
4. From the **Catalog\Cube** drop-down list, select the MDX cube to create metrics for.
5. Choose **Edit** > **Add New Compound Metric**.
6. Enter your expression in the **Definition pane** of the Metric Editor. For example, you can enter expressions similar to the following:

   - `([Discount Amount] * ?constantprompt)`

     This expression applies a special discount amount, entered by the user running the report. In this example, it is assumed that `constantprompt` is the name of a value prompt in the project and Discount Amount is a metric within the MDX cube.

   - `ApplySimple("([Measures].[Revenue],#0)", ?CategoryElementPrompt)`

     This expression creates a Revenue metric, which is conditional on an element list prompt answered by the user running the report. Users can then choose to view revenue data for different categories such as Books or Music. In this example, it is assumed that `CategoryElementPrompt` is the name of an element list prompt in the project that references a Category attribute within the MDX cube.
7. Click **Save and Close**.
8. In the **Object name** text field, enter a name for your metric.
9. Click **Save**.
10. Click **Save and Close**.

Deleting Compound Metrics from MDX Cubes

When you delete metrics based on multiple metrics of an MDX cube, dependencies may need to be resolved before you can delete the metric. When you try to delete a metric with dependent metrics, a list of metrics that are dependent on the metric you are deleting is returned. If a compound metric of an MDX cube has been added to any reports, a list of reports that depend on the metric is also returned. You must delete all of the metrics and reports which depend on the metric you are trying to delete. Then you can delete the metric.

You can remove the compound metric from the report rather than deleting the report. This removes the dependency between the metric and the report, and you can then remove the metric from the MDX cube.

For example, you import an MDX cube, and its data is automatically mapped to MicroStrategy metrics. You then create a new compound metric named Profit within the MDX cube by subtracting the MDX cube's cost data from its revenue data. Once this metric is created in your MDX cube, you create a Profit Margin metric that uses the Profit metric you just created. This makes the Profit Margin metric dependent on the Profit metric.

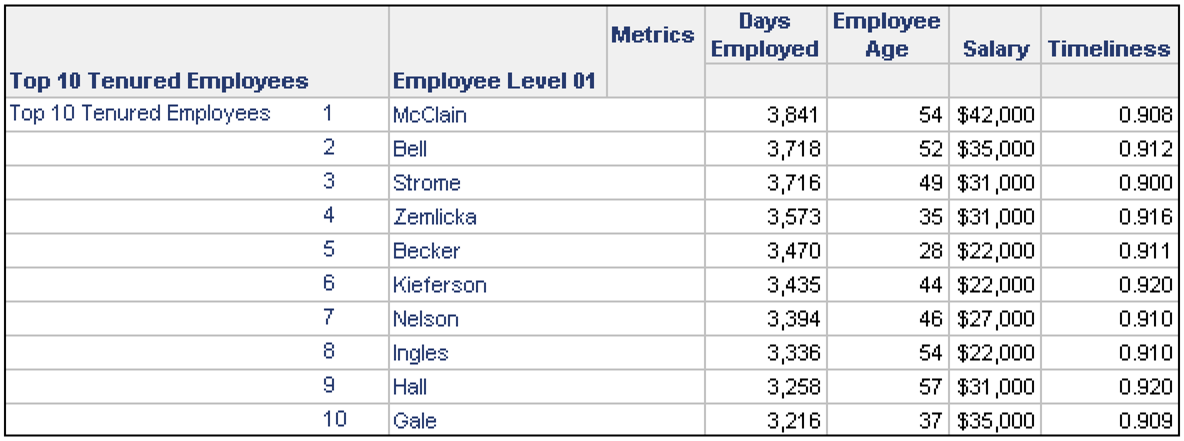
If you try to delete the Profit metric, a search for dependent objects is automatically triggered, and the Profit Margin metric is returned. To delete the Profit metric, you must first delete the Profit Margin metric. You also need to delete any reports that include the Profit metric or remove the Profit metric from the reports before you can delete the Profit metric from the MDX cube.

Creating Data Marts of MDX Cube Data

A MicroStrategy data mart is a data repository where you store the results of a report as a relational table in a data warehouse. After creating a data mart, you can use it as a source table in your projects, and execute reports against it.

Data marts can also be created based on MDX cube reports. Once this MDX cube data is available as data mart tables in your relational data warehouse, you can then map the data to the schema objects of your project, and then create standard MicroStrategy reports based on the data. This lets you take advantage of MicroStrategy features that cannot be used directly on MDX cube reports. These features include, but are not limited, to custom groups, consolidations, stand-alone filters, and metrics with complex definitions.

For example, the report below is based on MDX cube data from an SAP BW MDX cube source, which has been included in a project as a data mart. The report also includes the Top 10 Tenured Employees custom group, which could not be used directly on an MDX cube report. However, by including the MDX cube data in the project as a data mart and mapping the data to standard attributes and metrics, you can create a standard report based on this data that can take advantage of MicroStrategy features such as custom groups.



			Metrics	Days Employed	Employee Age	Salary	Timeliness
Top 10 Tenured Employees		Employee Level 01					
Top 10 Tenured Employees	1	McClain		3,841	54	$42,000	0.908
	2	Bell		3,718	52	$35,000	0.912
	3	Strome		3,716	49	$31,000	0.900
	4	Zemlicka		3,573	35	$31,000	0.916
	5	Becker		3,470	28	$22,000	0.911
	6	Kieferson		3,435	44	$22,000	0.920
	7	Nelson		3,394	46	$27,000	0.910
	8	Ingles		3,336	54	$22,000	0.910
	9	Hall		3,258	57	$31,000	0.920
	10	Gale		3,216	37	$35,000	0.909

Once you create an MDX cube report, you can use it to create a data mart. With the MDX cube report open in the Report Editor, from the **Data** menu, choose **Configure Data Mart**. For information on data marts, including prerequisites, examples, and steps to create a data mart, see the Advanced Reporting Help.

Reporting on MDX Cubes

A report in MicroStrategy is the central focus for MicroStrategy users to query, analyze, and visually present data in a manner that answers and evaluates their business questions. MDX cube reports provide the same data display and analysis functionality, but rather than reporting on data from a relational data warehouse, MDX cube reports report on data from MDX cube sources.

Before you can create an MDX cube report, you must import your MDX cubes and integrate them into a MicroStrategy project. For information on integrating MDX cube sources with MicroStrategy, see *Chapter 3, Integrating MDX Cubes into MicroStrategy*.

This section discusses the following topics related to MDX cube reports:

Create an MDX Cube Report

You can create an MDX cube report using an imported MDX cube as the data source. MDX cube reports can be created in either MicroStrategy Developer or Web.

The topics in this section explain options for creating and managing MDX cube reports:

With OLAP Services features, you can also report on MDX cubes by creating Intelligent Cube reports that connect to Intelligent Cubes containing MDX cube data. For steps to create an Intelligent Cube based on an MDX cube, see *Creating Intelligent Cubes Cased on MDX Cubes , page 147*. For steps to create an Intelligent Cube report, refer to the *In-memory Analytics Help*.

With MultiSource Option features, you can include MDX cube data along with data from your relational project. For steps to create a report with these analysis capabilities, see *Including MDX Cube Data in Standard Reports, page 149*.

Create an MDX Cube Report in Developer

To create an MDX cube report in Developer, you need to have Developer privileges, including the Define MDX Cube Report privilege.

At least one MDX cube must be imported into your MicroStrategy project. Importing MDX cubes is often handled by a MicroStrategy architect. For more information on importing MDX cubes into MicroStrategy, see *Importing MDX Cubes, page 83*.

You should have some experience and knowledge of designing MicroStrategy reports, as described in the Basic Reporting Help and the Advanced Reporting Help.

To Create an MDX Cube Report in Developer

1. Log in to a project connected to an MDX cube source.
2. Choose **File** > **New** > **Report**.
3. On the **MDX Sources** tab, choose a database instance connected to an MDX cube source and click **OK**.
4. Choose an MDX cube to report on:

   To show the technical names for MDX cubes, choose the **Display technical names** check box.

   - If MDX cubes have already been imported by an architect, they are displayed in their respective catalog structure. Click the plus sign (+) next to the catalog name to display its contents, and then choose the imported cube to use for your report.

  You can use the Find dialog box to search for a specific cube to use for your report.

- If the MDX cube that you want to report on is not imported yet, you can click **Retrieve cubes**. Choose the MDX cube from the list of MDX cubes from the chosen database instance. If you choose an MDX cube that has not been imported yet, it is imported into MicroStrategy before the report is created. You must have the Import MDX Cube privilege to perform this action.

  You can also use the MDX Cube Catalog to import cubes, as described in *Importing MDX Cubes, page 83*.

- You can later switch the MDX cube for your report if necessary, as described in *Switching the MDX Cube for an MDX Cube Report, page 151*.

5. Click **OK**.

Create an MDX Cube Report in MicroStrategy Web

To create an MDX cube report in Web, you need to have Web Professional privileges, including the Web Define MDX Cube Report privilege.

At least one MDX cube must be imported into your MicroStrategy project. Importing MDX cubes is often handled by a MicroStrategy architect. For more information on importing MDX cubes into MicroStrategy, see *Importing MDX Cubes, page 83*.

You should have some experience and knowledge of designing MicroStrategy reports, as described in the Basic Reporting Help and the Advanced Reporting Help.

To Create an MDX Cube Report in MicroStrategy Web

1. Log in to a project connected to an MDX cube source.
2. Click the **MicroStrategy** icon, and then choose **Create Report**.

3. From **Choose datasource**, choose **MDX Cube Report**.
4. Click an MDX cube source.
5. Click a catalog.
6. Choose an MDX cube.

   You can later switch the MDX cube for your report if necessary, as described in *Switching the MDX Cube for an MDX Cube Report, page 151*.

7. Click **OK**.

To Edit the MDX Cube Report Using the Report Editor

1. Choose attributes, metrics, prompts, hierarchies, and other objects as required from the Object Browser and put them on the report template. Some MicroStrategy objects on MDX cube reports work differently than on standard MicroStrategy reports. See *Hierarchies on MDX Cube Reports* and *Prompts on MDX Cube Reports* for more information.
2. Create a filter if needed. Filters on MDX cube reports work differently than on standard MicroStrategy reports, as described in the section *Filters on MDX Cube Reports, page 157*.
3. Choose **Data** > **Run Report**.

   From the **View** menu, you can change the view to Grid View, Graph View, Grid Graph View, or MDX View.

4. Format the report.

   You can use OLAP Services features such as view filters, derived metrics, and dynamic aggregation. If you are using MicroStrategy Web to create and view the MDX cube report, you can also use the OLAP Services feature derived elements to analyze the data. For details on how to use OLAP Services features, see the In-memory Analytics Help.

5. Click **Save and close**.

Troubleshooting MDX Cube Report Execution

The following are common issues for MDX cube report execution, and possible resolutions:

- If you experience long wait times when running MDX cube reports, this may be caused by the processing time required to load MDX cube schemas.

  MDX cube schemas that are loaded at report runtime can negatively affect the performance of MDX cube report execution. A project administrator can choose to load MDX cube schemas when Intelligence Server starts, thus removing the overhead of MDX cube schema loading from the report execution process. Steps to change the MDX cube schema load time are provided in *MDX Cube Schema Loading, page 67*.

- An MDX cube report fails with an error message that identifies data that has been mapped to an incompatible data type.

  A project designer can resolve the incompatible data type mapping to MDX cube data. Steps to resolve this type of error are provided in *About Defining Column Data Types for MDX Cube Data, page 112*.

- An MDX cube report fails with an error message that indicates the maximum row limit has been exceeded.

  A project administrator can resolve this error by increasing a project governing setting. Steps to increase the maximum row limit for reports are provided in *Supporting Large Result Sets for MDX Cube Reports, page 70*.

- A report that includes MDX cube data and relational project data fails with an error message that indicates data for an MDX metric is not available at the level for the report.

  This can occur if you are using MultiSource Option to include both MDX cube data and relational project data on a standard report. You can resolve this issue by removing any attributes from the report that are not mapped to data in the MDX cube used in the report. Creating these types of reports is described in *Including MDX Cube Data in Standard Reports, page 149*.

- An MDX cube report does not provide options to search the attribute elements for the report, or these options are greyed out, such as the ability to search for attribute elements can be provided when answering a prompt while executing a report or while selecting attributes as part of designing a report.

  These search options are unavailable if the MDX cube source the MDX cube report is associated with includes required variables. To allow attribute element searches on MDX cube reports for SAP BW cubes that include variables, within your SAP BW system, you must either define the variables as optional, or provide a default answer for the required variables. The MicroStrategy prompt that is mapped to the SAP BW variable can be either optional or required.

Creating Intelligent Cubes Cased on MDX Cubes

Once MDX cubes are integrated into MicroStrategy, you can begin to create MDX cube reports directly on these MDX cubes.

Alternatively, once an MDX cube is created, you can create an Intelligent Cube based on the MDX cube. This stores the MDX cube data as an Intelligent Cube, which allows you to take advantage of various Intelligent Cube features, including the improved response time of reporting against Intelligent Cubes.

Once an Intelligent Cube is created for an MDX cube, reports can be created based on the Intelligent Cube. These reports can analyze the MDX cube data, while also taking advantage of OLAP Services analysis features such as derived elements.

✓ You need the Use Intelligent Cube Editor privilege to create Intelligent Cubes, which is part of OLAP Services privileges.

To Create an Intelligent Cube Based on an MDX Cube

1. Using MicroStrategy Developer, log in to a project connected to an MDX cube source.
2. Choose **File** > **New** > **Intelligent Cube**.
3. On the **MDX Sources** tab, choose a database instance connected to an MDX cube source and click **OK**.
4. Choose an MDX cube to report on:

   To show the technical names for MDX cubes, choose the **Display technical names** check box.

   If MDX cubes have already been imported by an architect, they are displayed in their respective catalog structure. Click the plus sign (+) next to the catalog name to display its contents, and then choose the imported cube to use for your Intelligent Cube.

   or

   If the MDX cube that you want to report on has not been imported yet, click **Retrieve cubes**. Choose the MDX cube from the list of MDX cubes from the chosen database instance. If you choose an MDX cube that has not been imported yet, it is imported into MicroStrategy before the report is created. You must have the Import MDX Cube privilege to perform this action.

   You can also use the MDX Cube Catalog to import cubes, as described in *Importing MDX Cubes, page 83*.
5. Click **OK**.
6. From the **Object Browser** pane, choose attributes, metrics, and other objects from the MDX cube to create the data available for the Intelligent Cube.
7. Choose **File** > **Save**.

8. In the **Object name** field, type a descriptive name for the Intelligent Cube.

9. Click **OK**.

10. To make the Intelligent Cube available for creating reports, publish the Intelligent Cube by choosing **Run Report** in the toolbar.

Including MDX Cube Data in Standard Reports

In addition to creating MDX cube reports that report on single MDX cubes, you can include MDX cube data in standard reports so that both the relational project metrics and the MDX cube metrics on the same standard report.

✓ To create an MDX cube report in Developer, you need to have Developer privileges, including the Define MDX Cube Report privilege.

To create an MDX cube report in Web, you need to have Web Professional privileges, including the Web Define MDX Cube Report privilege.

At least one MDX cube must be imported into your MicroStrategy project. Importing MDX cubes is often handled by a MicroStrategy architect. For more information on importing MDX cubes into MicroStrategy, see *Importing MDX Cubes, page 83*.

To include MDX cube data in standard reports, you must map MDX cube columns for each project attribute you plan to include in the reports. For example, if a report includes the project attributes Year, Region, and Category, you must map MDX cube columns to these three attributes. For steps to map MDX cube columns to project attributes, see *Mapping MDX Cube Data to Project Attributes, page 106*.

You must have MultiSource Option privileges. For information on the capabilities available in MicroStrategy with the MultiSource Option, see the Project Design Help.

To include MDX cube data in standard reports

1. Create a new report by logging in to a project connected to an MDX cube source and:

   In Developer:

   a. Choose **File** > **New** > **Report**.

   b. Choose **General** > **Blank Report** > **OK**.

   In Web:

   a. Click the **MicroStrategy** icon and choose **Create Report**.

   b. From **Choose datasource**, choose **Blank Report**.

2. Select the attributes to include on the report. Attributes from the relational project can be added by navigating the folders of the project. You can include attributes that are also mapped to data for the MDX cube that you plan to report on.

   If you include attributes that are not on the MDX cube, an error is shown when trying to run the report that explains the data is not available at the specified level.

3. Choose the metrics to include on the report:

   - Metrics from the relational project can be added by navigating the folders of the project.
   - To include metrics from MDX cubes:
     - In Developer: From the **Object Browser** drop-down list, select the Data Explorer for an MDX cube source. The MDX cubes for the MDX cube source are displayed. Navigate to an MDX cube. Within an MDX cube, click the **Metrics** folder to view the metrics for an MDX cube. You can then drag and drop the MDX cube metrics onto the report.

- In Web: In the pane on the left, click **MDX Objects**. MDX cube data sources are displayed. Click the links for the MDX cube sources to navigate to an MDX cube. Within an MDX cube, click the **Metrics** folder to view the metrics for an MDX cube. You can then drag and drop the MDX cube metrics onto the report.

4. Include any other additional objects, such as prompts or filters, as required.
5. Run the report.

Switching the MDX Cube for an MDX Cube Report

After creating an MDX cube report, you can switch the MDX cube that is used as the source of data for the report. This can be helpful if another MDX cube has other additional or updated attribute information that could be of importance to the report.

The attributes that are included on the template of the MDX cube report must also be available in the MDX cube you are switching to. If the MDX cubes are from the same MDX source, the attributes are shared between the MDX cubes by default, as described in *Shared MDX Cube Objects, page 104*. However, if you have manually mapped the attributes in either of the MDX cubes to other attributes, either in the MDX cube source or in the relational project, the attributes in the two MDX cubes need to be mapped to the same objects to be able to switch between MDX cubes for the report.

The metrics that are included on the template of the MDX cube report must also be available in the MDX cube you are switching to. In addition, you must ensure that the metrics are shared between the two MDX cubes, as described in *Sharing metrics between MDX cubes, page 105*.

To switch the MDX cube for an MDX cube report

1. In Developer, log in to a project.
2. Browse to an MDX cube report, right-click the report and choose **Edit**.

3. Choose **Data** > **MDX Cube Options** > **Switch MDX Cube**.
4. Navigate to the MDX cube to use as the new source of data for the report.

   Only MDX cubes that meet the requirements described in the prerequisites of this procedure are available for selection. If no MDX cubes are available, you cannot switch the MDX cube for the report until you modify the MDX cubes as necessary.

5. Select the MDX cube and click **Open**.

Analyzing Data with MDX Cube Reports

MDX cube reports benefit from much of the same reporting and analysis functionality available for standard MicroStrategy reports. This section discusses the following reporting and analysis features, which have some unique functionality for MDX cube reports:

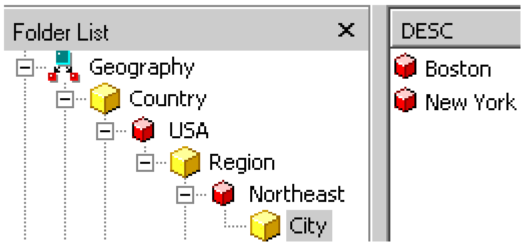
Hierarchies on MDX Cube Reports

You can include hierarchies of an MDX cube on the templates of MDX cube reports. Templates define the columns of data and data formatting displayed on reports and MDX cube reports. Hierarchies can be added to report templates using the same standard techniques to add attributes to a report template. At report run-time, any hierarchies included on an MDX cube report display the attributes that are part of that hierarchy.

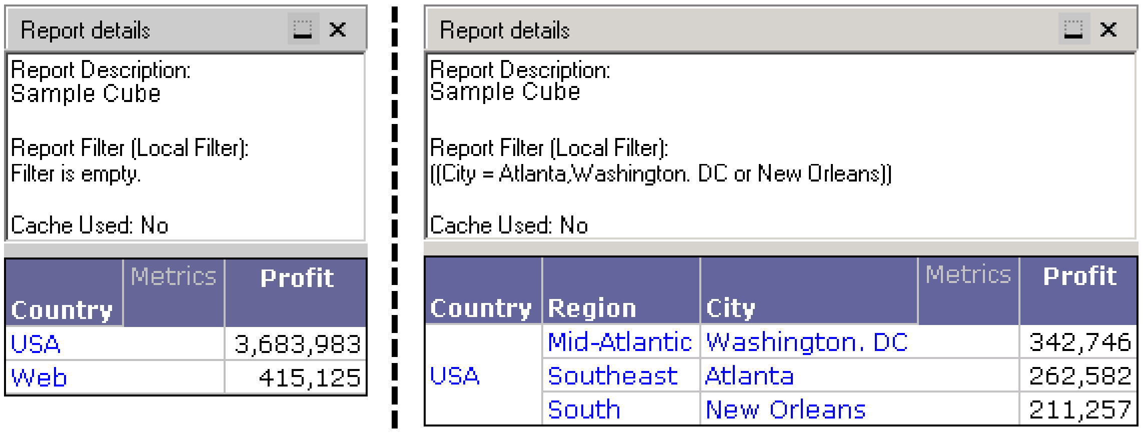
When an MDX cube report includes a hierarchy, the attributes that are displayed for that hierarchy are determined by two factors:

- The default number of attributes to display for a hierarchy. This is defined for a hierarchy when a project designer maps data from MDX cube sources to an MDX cube. For a detailed explanation of how to define the default number of attributes to display for a hierarchy, see *Displaying Hierarchies* *on MDX Cube Reports, page 124*.

- The attribute level defined in the report filter of the report. You can include an attribute in a report filter that is at a lower level than is set for its hierarchy. For example, you have a Geography hierarchy with Country, Region, and City attribute levels.



The lower attribute level of the hierarchy default and the report filter is used as the attribute level displayed on the report. In this hierarchy example, the hierarchy is defined to display one attribute on reports. If the Geography hierarchy is included on a report with no report filter qualifications on attributes of the Geography hierarchy, only Country is displayed for the hierarchy on the report. However, a report that includes a report filter qualification on City requires all attributes down to the level of the report filter qualification to be displayed for the hierarchy. (A qualification is the condition in a filter that limits the data to be included on a report.)


Report details

Report Description:
Sample Cube

Report Filter (Local Filter):
Filter is empty.

Cache Used: No

Country	Metrics	Profit
USA		3,683,983
Web		415,125

Report details

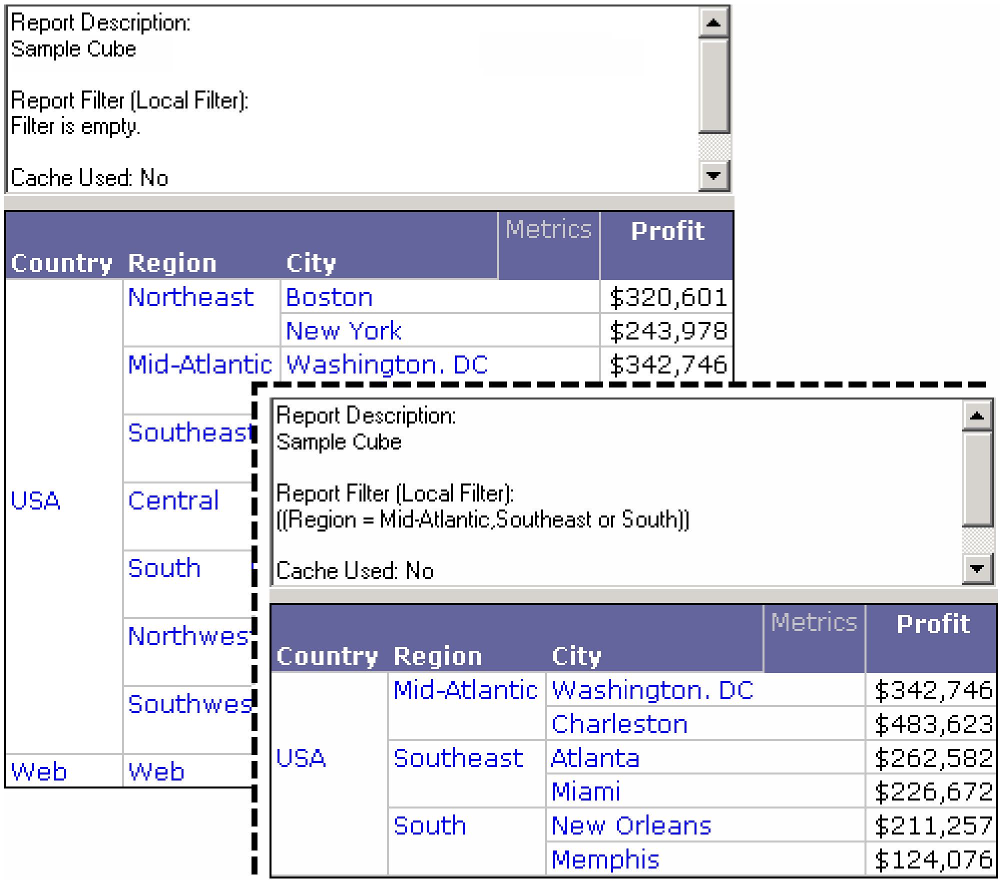
Report Description:
Sample Cube

Report Filter (Local Filter):
((City = Atlanta,Washington. DC or New Orleans))

Cache Used: No

Country	Region	City	Metrics	Profit
USA	Mid-Atlantic	Washington. DC		342,746
	Southeast	Atlanta		262,582
	South	New Orleans		211,257

The two example reports shown below illustrate the other scenario in which the Geography hierarchy has been defined to display all of its attributes on a report with a report filter at a higher attribute level. Even though the second report has a report filter on the Region attribute, all of the Geography attributes are displayed because the hierarchy is defined to display all attributes down to City.



Report Description:
Sample Cube

Report Filter (Local Filter):
Filter is empty.

Cache Used: No

Country	Region	City	Metrics Profit
USA	Northeast	Boston	$320,601
		New York	$243,978
	Mid-Atlantic	Washington, DC	$342,746
	Southeast		
	Central		
	South		
	Northwes		
	Southwes		
Web	Web		

Report Description:
Sample Cube

Report Filter (Local Filter):
((Region = Mid-Atlantic,Southeast or South))

Cache Used: No

Country	Region	City	Metrics Profit
USA	Mid-Atlantic	Washington, DC	$342,746
	Southeast	Charleston	$483,623
		Atlanta	$262,582
		Miami	$226,672
	South	New Orleans	$211,257
		Memphis	$124,076

When using hierarchies on MDX cube reports:

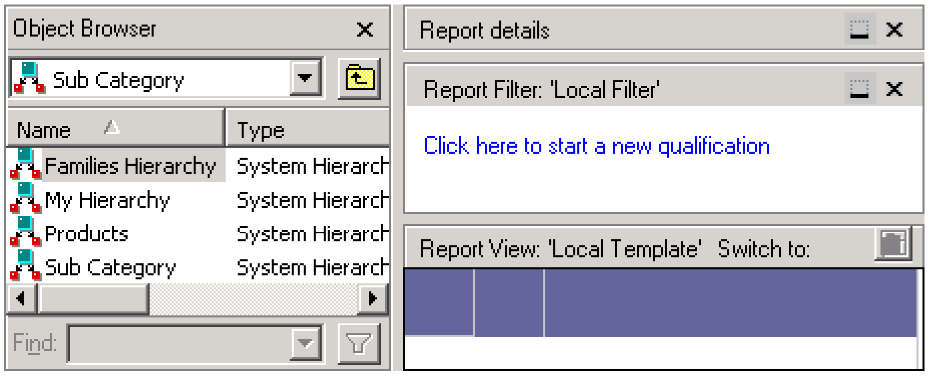
- Only the default report display forms for each attribute are shown when attributes are displayed as part of a hierarchy on a report. To modify the attribute forms that are displayed for each attribute when displayed as part of a hierarchy, you must modify the report display forms for the attributes using the Attribute Editor. For details on defining report display forms, see the Project Design Help.
- Attributes displayed as part of a hierarchy on a report act as one unit on the grid or graph. This means that all attributes displayed as a hierarchy

are pivoted or moved together. However, you can sort each attribute of a hierarchy individually by using the advanced sorting options for a report.

- Hierarchies must display attributes sequentially from the highest level attribute down to the lowest level defined for the hierarchy. This means that to see the lowest level attribute for a hierarchy displayed on the report, the hierarchy must display every attribute for the hierarchy. To see lower level attributes in the hierarchy without displaying the higher level attributes, add the attributes to the report one-by-one rather than including the hierarchy on the report.
- In SAP BW systems, dimensions can include multiple hierarchies for data display purposes. In MicroStrategy, only one hierarchy per dimension can be represented on an MDX cube report.
- You can drill on hierarchies included on your MDX cube reports using most of the standard techniques available for drilling on attributes on reports. For more information and best practices on drilling on hierarchies, see *Drilling on MDX Cube Reports, page 170*.

SAP BW dimensions with multiple hierarchies on reports

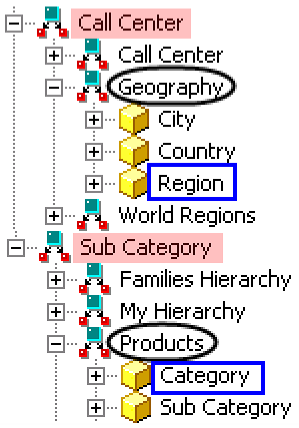
When data from an SAP BW MDX cube source is mapped to MicroStrategy objects, dimensions in SAP BW are mapped to a hierarchy object in MicroStrategy. However, some SAP BW dimensions are mapped to multiple hierarchy objects in MicroStrategy. This occurs because an SAP BW dimension can be broken up into separate MicroStrategy hierarchies for the purposes of formatting and structuring the display of data. For example, in the image shown below, the Sub Category dimension is broken up into four distinct MicroStrategy hierarchies; each hierarchy structures the same data in different ways.


Object Browser
Sub Category
Report details
Report Filter: 'Local Filter'
Click here to start a new qualification
Report View: 'Local Template' Switch to:
Find:

Name	Type
Families Hierarchy	System Hierarch
My Hierarchy	System Hierarch
Products	System Hierarch
Sub Category	System Hierarch

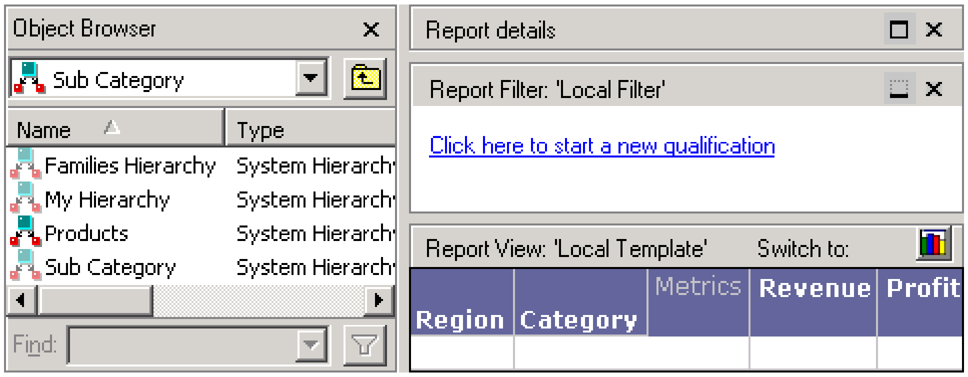
When including hierarchies or attributes from these hierarchies on MicroStrategy reports, you are restricted to including only one hierarchy per dimension. This means that once you add a hierarchy itself or attributes from a hierarchy on a report, all other hierarchies and their related attributes within the same dimension cannot be added to the report or report filter. This restriction also applies to prompts that contain objects from a different hierarchy within a dimension already included on the report. However, you can include other attributes, hierarchies, and prompts on a report or report filter as long as the hierarchies associated with these objects are not from the same dimension of the MDX cube.

For example, you have an SAP BW MDX cube with a Geography hierarchy within the Call Center dimension and a Products hierarchy within the Sub Category dimension.



You include the Region attribute from the Geography hierarchy and the Category attribute from the Products hierarchy on the report.

In the report below that once you include the Category attribute from the Products hierarchy, the rest of the hierarchies within the same dimension are made unavailable. This is because for each dimension you can only include a single hierarchy on a MicroStrategy report's template and filter. The report below also shows that attributes (in this example, Region and Category) from different hierarchies can be included on the same report as long as the hierarchies are in different dimensions.



Name	Type
Families Hierarchy	System Hierarch
My Hierarchy	System Hierarch
Products	System Hierarch
Sub Category	System Hierarch

Report Filter: 'Local Filter'

Click here to start a new qualification

Region	Category	Revenue	Profit

You could also include the Geography and Products hierarchy objects on the same report together rather than choosing individual attributes. This is possible because the hierarchies are from different dimensions.

Filters on MDX Cube Reports

In MicroStrategy you can use a filter to specify the conditions that report data must meet to be included in report results. For MDX Cube reports, most of the filtering features remain the same as those for standard MicroStrategy reports. For more general information about filtering, refer to the Filters section in Advanced Reporting Help.

In a standard report, the filter is evaluated independently of the report template in most cases. However, in an MDX cube report, due to the nature of MDX, a close relationship exists between the objects in the filter and the objects in the report template. Because of this relationship, qualifications on MDX cube reports are performed at different points during report execution:

- Qualifications on dimensions that are not included in the template are evaluated as a filter before metric aggregation. For example, the Year attribute is qualified on in the report filter and an attribute in a different dimension such as Store is on the template. In this scenario, each metric value is restricted to consider only those years determined by the filter before aggregation.
- Qualifications on dimensions that are included in the template are applied as a limit after metric aggregation. For example, the Year attribute is on the template and Year is qualified on in the report filter. In this scenario, the report results are restricted as a limit after metric aggregation.

Metric qualifications that have a specific output level are evaluated along with the output level attribute, either before or after aggregation.

The logical relationships between qualifications are set automatically depending on the dimension or dimensions to which the filtered objects belong. The following rules govern the logical operators between qualifications:

- Qualifications that define attributes in the same dimension are joined by the OR operator. For example, Customer and Customer Region both belong to the Customer dimension and would be joined with the OR operator.
- Qualifications that define attributes in different dimensions are joined by the AND operator. For example, Category belongs to the Product dimension, and Year belongs to the Time dimension and would be joined with the AND operator.
- Metric limits are always joined by the AND operator with all the other qualifications.

If you plan to perform data qualifications on MDX property data, you can support mapping MicroStrategy date forms to MDX property data of the date data type. For information on how to support these date forms and qualifications, see *Supporting MDX Cube Source Date Data in MicroStrategy, page 120*.

See the following to learn more about filtering options for MDX cubes:

Specifications for Report Filter Qualifications

The workflow to create a report filter in an MDX cube report is slightly different than creating a report filter for a standard MicroStrategy report. Rather than creating a report filter qualification by selecting the type of qualification, in MDX cube reports you create report filter qualifications by selecting the fields, operators, and values for the qualification, such as:

- **Field**: The field of a qualification is the object that you are qualifying on. You can only choose attributes and metrics which are included in the MDX cube that the MDX cube report is connected to. The type of object you choose determines the type of qualification you create, as follows:
  - **Attribute**: Selecting an attribute available in the MDX cube enables you to create an attribute qualification, or a metric set qualification with the selected attribute as the set qualification's output level. For attribute qualifications, you can create an element list qualification by selecting the In list or Not in list operator, or you can qualify on any available attribute forms by selecting the Where operator.

    Due to MDX restrictions, the MDX cube data that is automatically mapped to the DESC attribute form in MicroStrategy cannot be used in attribute qualifications. To support attribute qualifications on the MicroStrategy DESC attribute form, you must map the DESC form to one of the extended properties of the level in the MDX cube. This mapping can be done in the MDX Cube Catalog (see *Mapping MDX Cubes, page 99*).

  - **Metric**: Selecting a metric available in the MDX cube enables you to create a metric set qualification. To create a metric set qualification with a specific attribute level, you must choose an attribute as the first field. Metric set qualifications filter report data based on metric data restrictions.

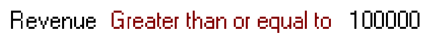
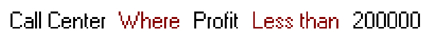
- **Operator**: The operator of the qualification determines how to qualify the object of the qualification (field) on the values you have selected. The operators available depend on the object of the qualification. For example, if you are qualifying on an attribute, you can choose from the operators In list, Not in list, and Where. Qualifying on a metric enables you to select from various comparison operators.

  SAP BW does not support string operators. Therefore, operators such as `LIKE`, `CONTAINS`, and so on cannot be used for SAP BW MDX cubes.

- **Value**: The values combined with the operator comprise the qualification criterion used to filter the report. The value of the qualification can be a list of attribute elements (for attribute element list qualifications), a list of values, a single value, or a prompt that returns a value.

This table lists the types of filter qualifications available for MDX cube reports and the general format required to create them:

Filter Qualification Type	Format	Example
Attribute element list qualifications	**Field** = *attribute* **Operator** = In list Not in list **Value** = *list of attribute elements*	Call Center In list {Atlanta, San Diego}
Attribute form qualifications	**Field** = *attribute* **Operator** = Where	Category Where ID Between 2 AND 4

Filter Qualification Type	Format	Example
	Field = *attribute form* **Operator** = *any available operator* **Value** = *any valid value or value prompt*	
Metric set qualifications without an output level	**Field** = *metric* **Operator** = *any available operator* **Value** = *any valid value or value prompt*	 Revenue Greater than or equal to 100000
Metric set qualifications with an output level	**Field** = *attribute* **Operator** = Where **Field** = *metric* **Operator** = *any available operator* **Value** = *any valid value or value prompt*	 Call Center Where Profit Less than 200000

You can only use objects on an MDX cube report that are available within the associated MDX cube. Since report filter qualifications in MDX cube reports depend on the associated MDX cube, you cannot save the qualifications as filters for use with other reports.

Once you create a report filter qualification, you can convert it into a prompt to enable MDX cube report users to select their own filtering criteria. For information on converting report filter qualifications into prompts, see *Prompts on MDX Cube Reports, page 165*.

Create a Report Filter Qualification

The options you see to create a report filter qualification change depending on your choices for each component. See *Specifications for Report Filter Qualifications* for information on how field, operator, and value components define a report filter qualification.

1. *Create an MDX Cube Report*.

   or

   Open an MDX cube report from Developer by right-clicking the report and choosing **Edit**.

2. Add attributes and metrics to the MDX cube report.
3. Choose **Report Filter** > **Click here to start a new qualification**.
4. Click **Field**, and choose the attribute or metric to qualify on.

   If the MDX cube report is connected to an SAP BW MDX cube with dimensions that contain multiple hierarchies, you may only be able to choose from a subset of the attributes available in the MDX cube. This is because, within a dimension, only one hierarchy can be represented on the report and report filter. For more information on this restriction, see *SAP BW dimensions with multiple hierarchies on reports, page 155*.

5. Click **Operator** and choose the operator for the qualification.
6. Click **Value**. Either type a value, choose attribute elements, or choose a prompt that returns a value.

   If you need to import a text file for an attribute element qualification, note that the same rules apply to MDX cube reports as for standard reports. Namely, data in the file needs to be tab-delimited, return-delimited, or list-delimited as specified in the Regional Settings.

Filtering Metric Data at a Specific Attribute Level

The attribute level of a metric set qualification specifies the output level at which the metric is calculated for the qualification. For example, if a metric set qualification is Sales > 100000, Sales could mean sales per day, month, or year. You must specify a time attribute as an output level to clarify the qualification.

By default, metric set qualifications are evaluated using the level of the metric itself. However, metrics commonly do not have a defined level. In this case, metric set qualifications are evaluated at the report level, which is the level defined by the lowest-level attributes on the report. To return more targeted results, you can create a metric set qualification to produce a subset of the report results.

To create a metric set qualification with an output level

1. *Create an MDX Cube Report*.
2. Add attributes and metrics to the report.
3. Choose **Report Filter** > **Click here to start a new qualification**.
4. Click **Field** and choose the attribute to use as the output level for the metric set qualification.
5. Choose **Operator** > **Where**.
6. Click **Field** and select the metric to qualify on.
7. Click **Operator** and select the operator for the qualification.
8. Click **Value** and type a value or select a prompt that returns a value.
9. Run the report.

Filtering with Static or Dynamic Date Qualifications

You can create static and dynamic date qualifications in MicroStrategy to filter the data in your MDX cube reports. Static date qualifications use a specific date to qualify on data while dynamic date qualifications enable you to use conditions based on the current date to qualify on data. For example, a dynamic date can be used in a report that examines revenue amounts from the previous two months. This is represented as "today" with an offset of two months. For background information on and examples of dynamic date qualifications, see the Advanced Reporting Help.

Date data represents a given day using various formats that include the month, day, and year of a given day. In MicroStrategy, this type of data is commonly represented as an attribute form of an attribute that contains data describing individual days. Before you can filter data using date qualifications, the data of your MDX cube report must have some data defined as the date data type. For steps to define data in an MDX cube as the date data type, see *Supporting MDX Cube Source Date Data in MicroStrategy, page 120*.

To create a date qualification on MDX cube data

1. *Create an MDX Cube Report* connected to an MDX cube that contains a date attribute.
2. Drag-and-drop the attribute to create a date qualification for into the **Report Filter** pane. The attribute is included as the Field of a report filter qualification.
3. Choose **Operator** > **Where**.
4. Click **Field** and choose the attribute form that is defined with a date data type.
5. Click the new **Operator** option and choose the operator for the qualification.
6. Click **Value**and choose:

Use Calendar: This option enables you to select a static date from a calendar. You cannot create a dynamic date qualification with this option.

or

Select a Dynamic Date: You can select a static date from a calendar or create a dynamic date qualification.

Dynamic date qualifications are created as an offset of the current date. The Date Editor provides a preview of what your dynamic date qualification would return using the current date as the starting point. See the Advanced Reporting Help for more information about dynamic date qualifications.

or

Select a Prompt and **Prompt a Value**: These options enable you to select a date prompt or create a new date prompt for a user to answer at report runtime to complete the filter qualification. Prompts can use static or dynamic dates to return a list of answers for users to choose from.

If you do not see these date options, this means the attribute form has not been defined correctly as a date data type. For steps to define MDX cube data as a date data type, see *Supporting MDX Cube Source Date Data in MicroStrategy, page 120*.

7. Add attributes, metrics, and any other valid objects to meet your remaining report requirements.

8. Click **Save and Close**.

Prompts on MDX Cube Reports

MicroStrategy automatically converts SAP BW variables into prompts when SAP BW cubes are imported into a project. In addition to these inherited prompts, you can create standard prompts for MDX cube sources in the same way as you can in MicroStrategy reports that access a data warehouse.

To allow attribute element searches on MDX cube reports for SAP BW cubes that include variables, within your SAP BW system, you must either define the variables as optional, or provide a default answer for the required variables. The MicroStrategy prompt that is mapped to the SAP BW variable can be either optional or required.

You can display prompt details for MDX cube reports just as you can for standard reports by opening the Developer Preferences dialog, choosing **Reports**, **Show report details**, and then **Include prompt details**. This feature is especially useful if you want to display a summary of the variable elements that are used to answer the variable prompts.

There are two ways to create a standard prompt for an MDX cube report. You can convert report filters to prompts or create new prompts using the Prompt Generation Wizard for supported prompt types:

You can select attributes and metrics for your prompt definitions by browsing to the **Data Explorer** for your MDX cube source.

When you save a prompted report, whether it has inherited prompts converted from SAP BW variables or standard prompts, you can to save it as a static or prompted report. See the Basic Reporting Help for details on saving a prompted report.

Convert Report Filters to Prompts

To convert report filters to prompts:

1. Create a report filter qualification for an MDX cube report.
2. Right-click the filter qualification and choosing **Convert to Prompt**.

When a report filter qualification is converted into a filter definition prompt, the type of filter definition prompt that is created depends on the type of filter qualification you convert. The following table below lists the prompt type created for different types of filter qualifications.

Filter Qualification Type	Filter Definition Prompt Type Created
Attribute element list qualifications	An attribute element list prompt, based on the attribute of the attribute element list qualification. Any attribute elements selected as values for the filter qualification are converted into default answers for the attribute element list prompt.
Attribute form qualifications	An attribute prompt, based on the attribute of the attribute form qualification. The attribute form qualification is converted into the default answer for the attribute prompt.
Metric set qualifications	A metric prompt, based on the metric of the metric set qualification. The metric set qualification is converted into the default answer for the metric prompt.

Create New Prompts Using the Prompt Generation Wizard

Using the Prompt Generation Wizard is the same process as creating a standard MicroStrategy prompt. Prompts in MDX cube reports can only access objects that are included in an MDX cube report's associated MDX cube. Therefore, when using the Prompt Generation Wizard to create prompts for an MDX cube report, it is important that the prompt only contain objects from the associated MDX cube.

For example, an MDX cube report is associated with MDX cube 1 that has metrics Revenue, Cost, and Profit. Including an object prompt with the Revenue, Cost, and Profit metrics from MDX cube 1 enables users of the report to choose which of these metrics they want to view on the report. However, if the object prompt included a metric from an MDX cube that is not associated with the MDX cube report, the prompt cannot be included in the MDX cube report. The same object prompt could be used in other MDX cube reports that are associated with MDX cube 1.

Supported Prompt Types

When you create a prompt for an MDX cube report using the Prompt Generation Wizard, you must select the type of prompt you want to create. While most of the standard prompt types are supported for MDX cube reports, there are some guidelines you should be aware of when creating prompts for your MDX cubes. This table lists the prompt types supported for MDX cube reports and guidelines on how to create them and include them in MDX cube reports:

Prompt Type	Guidelines For Creation
Filter definition prompt: Choose from all attributes in a hierarchy	Creating the prompt: You must choose a specific hierarchy rather than using the results of a search or listing all hierarchies with no restrictions. This ensures that objects included in the prompt all come from the same MDX cube. Including the prompt in an MDX cube report: When you include the prompt in the report filter of a report, it must meet the required logical relationships between filter qualifications described in *Filters on MDX Cube Reports, page 157*. MDX cube reports associated with an SAP BW MDX cube can only have one hierarchy per dimension on a report. Therefore, you can only add this type of prompt on an MDX cube report associated with an SAP BW MDX cube if it meets the one hierarchy per dimension requirement, which is described in *SAP BW dimensions with multiple hierarchies on reports, page 155*.
Filter definition prompt: Qualify on an attribute	Creating the prompt: You must choose a specific attribute rather than using the results of a search. This ensures that attributes included in the prompt all come from the same MDX cube. Including the prompt in an MDX cube report:

Prompt Type	Guidelines For Creation
	When you include the prompt in the report filter of a report, it must meet the required logical relationships between filter qualifications described in *Filters on MDX Cube Reports, page 157*.
Filter definition prompt: Choose from an attribute element list	Creating the prompt: To determine the attribute elements available as prompt answers, you must list all attribute elements or select a list of elements rather than use a filter. This ensures that the attribute elements included in the prompt all come from the same MDX cube. Including the prompt in an MDX cube report: When you include the prompt in the report filter of a report, it must meet the required logical relationships between filter qualifications described in *Filters on MDX Cube Reports, page 157*.
Filter definition prompt: Qualify on a metric	Creating the prompt: You must choose a specific metric rather than using the results of a search. This ensures that metrics included in the prompt all come from the same MDX cube. Including the prompt in an MDX cube report: When you include the prompt in the report filter of a report, it must meet the required logical relationships between filter qualifications described in *Filters on MDX Cube Reports, page 157*.
Object prompts	Creating the prompt: You must choose specific objects rather than using the results of a search. This ensures that objects included in the prompt all come from the same MDX cube.
Value prompts	Creating the prompt: You can create standard MicroStrategy value prompts to use

Prompt Type	Guidelines For Creation
	in MDX cube reports. Including the prompt in an MDX cube report: To use a value prompts with MDX cube data, you must define MDX cube data as a data type that matches a value prompt. For instructions on how to define column data types for MDX cube data, see *About Defining Column Data Types for MDX Cube Data, page 112*. For instructions on how to define data in an MDX cube as the date data type, see *Supporting MDX Cube Source Date Data in MicroStrategy, page 120*.

Drilling on MDX Cube Reports

You can drill in reports to obtain additional information after a report has been executed. For MDX cube reports, drilling is supported within the MDX cube. This means that you can drill up or down within the dimension that the attribute belongs to, as well as in other directions to dimensions included in the same MDX cube.

Drill paths are automatically generated based on the definition of the MDX cube. You cannot drill down into the elements of SAP BW structures. For more information on structures, see *Relating Objects from SAP BW to MicroStrategy, page 10*.

When running an MDX cube report in MicroStrategy Web, using the sub-menu display option to view your drilling options in MDX cube reports will allow you to drill in other directions to all available hierarchies within the MDX cube rather than only allowing you to drill up and down within the current hierarchy.

See the following for detailed information about how to drill on these reports:

Display All Available Hierarchies as Drilling Options in MicroStrategy Web

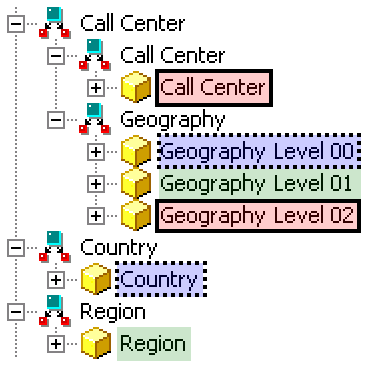
1. In MicroStrategy Web, log in to a project.
2. Click the **MicroStrategy** icon, and choose **Preferences**.
3. Choose **Preferences Level** > **Project Defaults**.
4. Choose **Preferences** > **Drill Mode**.
5. Choose **Display advanced drill options as** > **Sub menus on the context menu**.
6. Click **Apply**.

Available Drill Paths with MDX Cube Reports

In standard MicroStrategy reports, when you drill on attributes that are part of the relational project schema, drill paths enable you to drill up and down to other attributes that are part of the same hierarchy. The up and down drill paths are based on the system hierarchy. In MDX cube reports, you can drill across from any attribute to any other attribute available in the MDX cube, but up and down drill paths may not be available. The drill paths available in MDX cube reports are determined by whether attributes are mapped directly to characteristics, or whether attributes are mapped to attributes that are part of a hierarchy.

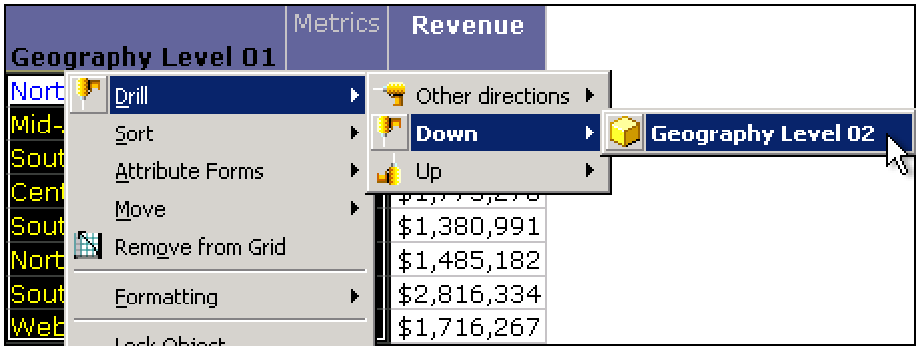
Up and down drill paths for an attribute are only available for attributes mapped to levels of an SAP BW hierarchy. In SAP BW, hierarchies can be created to organize and logically relate a group of characteristic data (represented as attributes in MicroStrategy). When these hierarchies are imported into MicroStrategy, up and down drill paths are created for the hierarchy's attributes.

For example, you create an MDX cube report on an SAP BW MDX cube. The MDX cube includes a Geography hierarchy which organizes the characteristics Country, Region, and Call Center (Geography Level 00, Level 01, and Level 02 respectively). Attributes in MicroStrategy are mapped to the characteristics Country, Region, and Call Center of the MDX cube.



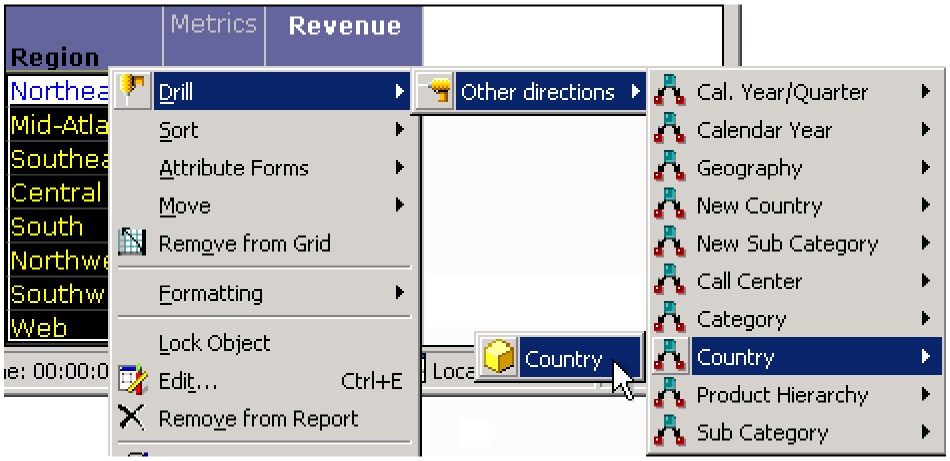
In this example, the attributes of the Geography hierarchy are intentionally not renamed so that the difference between the attributes that are part of the hierarchy and the attributes mapped directly to the characteristics can be observed. You can rename the hierarchy attributes to match the attributes mapped to the characteristics. For example, you can rename the attribute Geography Level 00 as Country.

You then create an MDX cube report that includes the Geography Level 01 attribute of the Geography hierarchy that contains regional data. To drill down to call center data, you can drill down on the Geography Level 01 attribute.



You can also drill up to country data or you can use the other drill path directions to drill to any other attribute available in the MDX cube.

You can create an MDX cube report that includes the Region attribute mapped directly to the Region characteristic in SAP BW. When you drill on this attribute, you can still drill to country data, but this is only available as an **Other directions** drill path.



To enable up and down drill paths on your MDX cube reports, you must include the attributes that are part of a hierarchy during the mapping process. Including attributes that are mapped directly to characteristics enables drilling to the same set of attributes available in the MDX cube, but all drilling is available only through the **Other directions** drill path.

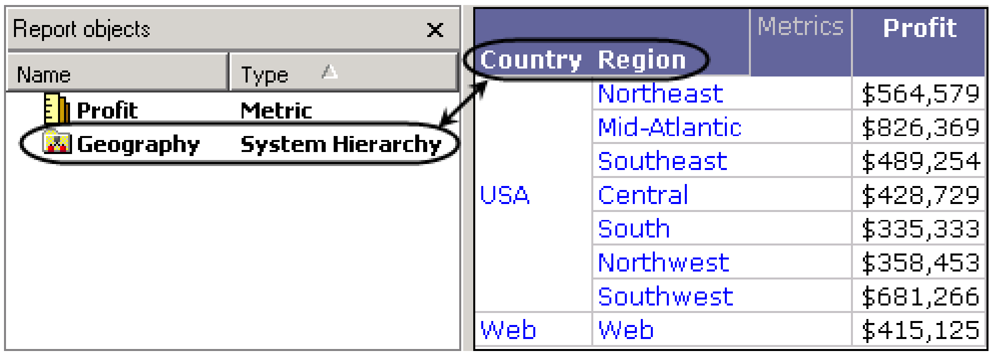
Drilling on Hierarchies in MDX Cube Reports

You can drill on hierarchies included on your MDX cube reports using most of the standard techniques available for drilling on attributes in MicroStrategy reports.

MicroStrategy recommends the following best practices for drilling on hierarchies in MDX cube reports:

- Drilling up to a higher-level attribute in a hierarchy is disabled. This is because the highest level attribute is always included on the report since hierarchies must display attributes sequentially from the highest level attribute down to the lowest level defined for the hierarchy.

- You can drill down from a hierarchy to any attribute within the hierarchy not already included on the report.
- You can drill from a hierarchy to attributes within different hierarchies as long as the hierarchy is in a different dimension. This is a requirement of SAP BW's definition of hierarchies and dimensions, which affects what hierarchies can be included on the same report. This is explained in more detail in *SAP BW dimensions with multiple hierarchies on reports, page 155*.
- If you choose to keep the hierarchy on the report you drill to, the hierarchy object is replaced with the attribute objects that were displayed for the hierarchy in the original report. For example, consider an MDX cube report that includes a Geography hierarchy with the attributes Country, Region, and City. In this example your original report includes Country and Region as part of the Geography hierarchy.

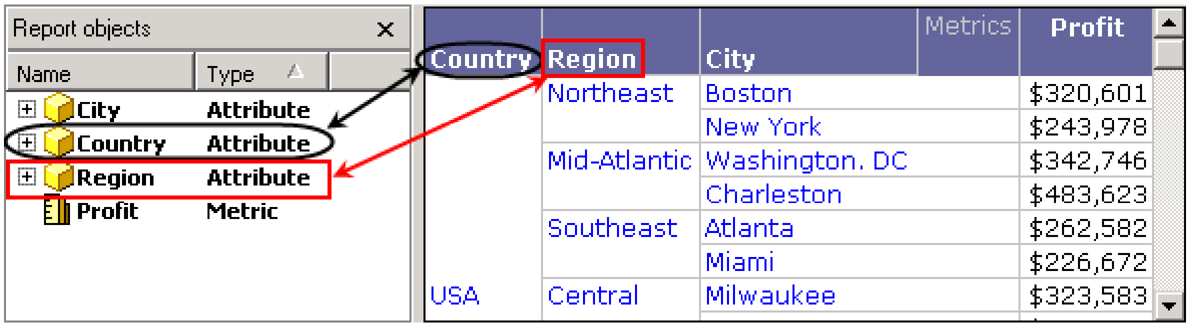


Report objects

Name	Type
Profit	Metric
Geography	System Hierarchy

Country	Region	Profit
USA	Northeast	$564,579
	Mid-Atlantic	$826,369
	Southeast	$489,254
	Central	$428,729
	South	$335,333
	Northwest	$358,453
	Southwest	$681,266
Web	Web	$415,125

When you drill down to City and choose to keep the hierarchy on the report you drill to, the Geography hierarchy is replaced with the Country and Region attributes that were displayed on the original report.



Report objects

Name	Type
City	Attribute
Country	Attribute
Region	Attribute
Profit	Metric

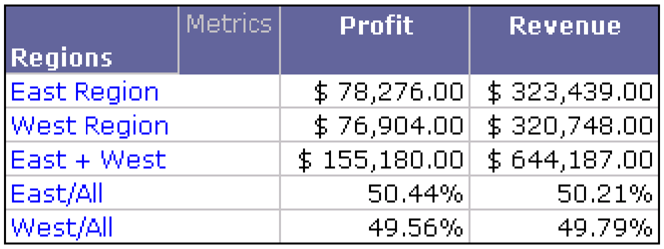
Country	Region	City	Profit
USA	Northeast	Boston	$320,601
		New York	$243,978
	Mid-Atlantic	Washington, DC	$342,746
		Charleston	$483,623
	Southeast	Atlanta	$262,582
		Miami	$226,672
	Central	Milwaukee	$323,583

Since the attributes are included in the drilled to report, you can then use them individually as you can use any attributes on reports. For example you can pivot them on the grid, or move them to the page-by area.

Sorting Structure Elements and Preserving Order

In SAP BW, when you create a characteristic structure and its structure elements, you can re-order the structure elements as required. When characteristic structures are imported into MicroStrategy as attributes, the order of the structure elements in your SAP BW MDX cube source is preserved as the default order of the attribute elements. This maintains a consistent view of your data across your SAP BW MDX cube source and MicroStrategy.

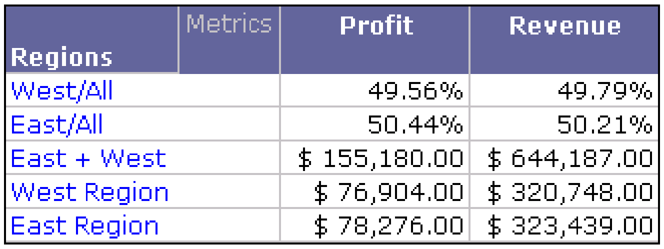
For example, the report shown below includes a Regions attribute mapped to a characteristic structure.



Regions	Metrics	Profit	Revenue
East Region		$ 78,276.00	$ 323,439.00
West Region		$ 76,904.00	$ 320,748.00
East + West		$ 155,180.00	$ 644,187.00
East/All		50.44%	50.21%
West/All		49.56%	49.79%

The attribute elements East Region, West Region, East + West, East/All, and West/All are displayed in the same order that is available in the SAP BW MDX cube source. If the order of structure elements has changed in your SAP BW MDX cube source, you can update the MDX cube structure to apply these changes in MicroStrategy (see *Updating MDX Cube Structure, page 94*).

You can also use this structure element order to sort the information on the report. For example, you can perform a descending sort on this structure element order to display the structure elements in the opposite order.



Regions	Metrics	Profit	Revenue
West/All		49.56%	49.79%
East/All		50.44%	50.21%
East + West		$ 155,180.00	$ 644,187.00
West Region		$ 76,904.00	$ 320,748.00
East Region		$ 78,276.00	$ 323,439.00

To sort on structure element orders

1. In Developer, log in to a project.
2. Browse to an MDX cube report, right-click and choose **Run**.
3. Choose **Data** > **Advanced Sorting**.
4. Choose **Rows** > **Add**.
5. Choose **Sort By**, and choose the attribute mapped to the structure.
6. Choose **Criteria** > **Source Order**.

   Source Order is an attribute form automatically created for the attribute mapped to an SAP BW structure to preserve the structure element order.

7. From the **Order** column, choose **Ascending** or **Descending**.
8. Click **OK**.
9. Re-execute the report.

Sorting on Attribute Element Orders from MDX Cube Sources

The order of MDX cube data mapped to attribute elements in MicroStrategy can be defined to include the same order of the data in your MDX cube source. If the order is defined in this way, as described in *Preserving Attribute Element Orders from MDX Cube Sources, page 117*, you can sort MDX cube reports so that the data is in the same order as it exists in the MDX cube source.

Sorting MDX cube reports to represent data as it exists in your MDX cube source can be helpful in various scenarios. For example, this can help support financial balance sheet analysis in which you always want to see your accounts receivable information above your accounts payable information.

⚠ You can only sort an MDX cube report based on the order in your MDX cube source if the order has been integrated into MicroStrategy. For information on defining MDX cubes to integrate their order of data into MicroStrategy, see *Preserving Attribute Element Orders from MDX Cube Sources, page 117*.

To sort on attribute element orders from MDX cube sources

1. Log in to a project in Developer.
2. Browse to an MDX cube report, right-click and choose **Run**.
3. Choose **Data** > **Advanced Sorting**.
4. Choose **Rows** > **Add**.
5. From the **Sort By** column, choose an attribute on the MDX cube report.

   If you are sorting multiple attributes in a hierarchy and want to replicate the order that exists in your MDX cube source, you should sort on higher level attributes first. For example, if your have attributes for Year, Quarter, and Day, you should first sort on Year, then on Quarter, and then on Day.
6. Choose **Criteria** > **Source Order**.

   Source Order is an attribute form automatically created for any attributes within dimensions that are defined to integrate the order of their data into MicroStrategy. If there is no Source Order form, the order information has not been integrated. For information on defining MDX cubes to integrate their order of data into MicroStrategy, see

Preserving Attribute Element Orders from MDX Cube Sources, page 117.

7. From the drop-down list in the **Order** column, choose **Ascending**.

   An ascending sort on the Source Order form displays the data in the same order that exists in the MDX cube source.

8. Click **OK**.

9. Re-execute the report.

Inheriting MDX Cube Source Formats for Metric Values

You can specify for the metric values in MicroStrategy MDX cube reports to inherit their value formatting from an MDX cube source. This enables MicroStrategy MDX cube reports to use the same data formatting available in your MDX cube source and maintains a consistent view of your MDX cube source data in MicroStrategy.

Inheriting value formats from your MDX cube source also enables you to apply multiple value formats to a single MicroStrategy metric. Normally, a MicroStrategy metric can only have one value format for all of its values.

Custom groups and consolidations can apply multiple value formats for a metric. However, custom groups and consolidations are not available for MDX cube reports. You can however use custom groups and consolidations on MDX cube data if you create a data mart from an MDX cube report. For information on creating data marts from MDX cube reports, see *Creating Data Marts of MDX Cube Data, page 140*.

For information on custom groups and consolidations, see in the Advanced Reporting Help.

Some MDX cube sources allow data that can be mapped to MicroStrategy metrics to have more than one value format, which is common when reports reflect the locale of the data or when they include characteristic structures mapped to MicroStrategy attributes, performing calculations that return various value formats such as percent and dollar amount.

To inherit MDX cube source formats for metric values

1. In Developer, log in to a project.

   VLDB properties cannot be modified from MicroStrategy Web.

2. Browse to an MDX cube report, right-click the report, and choose **Run**.
3. Choose **Data** > **VLDB Properties**.
4. Choose **Tools** > **Show Advanced Settings**.
5. In the **VLDB Settings** list, expand **MDX**, and choose **MDX Cell Formatting**.
6. Clear the **Use default inherited value** check box.
7. Choose one of the following options:

   MDX metric values are formatted per column: If you choose this option, MDX cube source formatting is not inherited. You can format each metric on the MDX cube report to use only one value format for all the metric values.

   or

   MDX metric values are formatted per cell: If you choose this option, MDX cube source formatting is inherited. Metric value formats are determined by the formatting that is available in the MDX cube source, and a metric can have various value formats.

   You can use standard MicroStrategy techniques to further format the inherited formatting for metric values. However, you must use default Number formatting. Modifying the Number formatting for the metric values to anything other than default removes any inherited formatting from the MDX cube source. Number formats include general, fixed, currency, percent, and so on.

8. Click **Save and Close**.

The MDX Cell Formatting VLDB property can also be defined for database instances connected to MDX cube sources. Defining this VLDB property for a database instance applies the definition as the default for all MDX cube reports associated with the database instance. For steps to define this VLDB property for a database instance, see *Defining MDX Cube Sources to Inherit Formats for Metric Values, page 71*.

Using MDX Cube Reports to Filter Other Reports

MDX cube reports can be used to filter other standard reports in MicroStrategy. This can be supported by using an MDX cube report as shortcut-to-a-report qualification on a standard report. For information on shortcut-to-a-report qualification, see the Advanced Reporting Help.

In Developer, with a standard report open, you choose **Add a Shortcut to a Report** to access the report as filter functionality.

The MDX cube report must map data to project attributes rather than managed object attributes. This is required for the standard report to recognize the data that is available on the MDX cube report. For information on mapping data in MDX cube reports to project attributes, see *Mapping MDX Cube Data to Project Attributes, page 106*.

You must have the Execute Multiple Source Report privilege, which is part of the MultiSource Option group of privileges. For information on MultiSource Option, see the Project Design Help.

To add an MDX cube report as a shortcut-to-a-report qualification in a standard report

1. Open the standard report in Developer.
2. In the **Report Filter** area, double-click the arrow to add a new qualification.
3. Choose **Filtering Options** > **Add a Shortcut to a Report**.
4. Click **OK**.
5. Click the browse button **(...)** to choose an MDX cube report.
6. Click **OK** twice.