Infor BI OLAP Server Administration Guide

Release 11.0.x

Publication Information

Release: Infor BI 11.0.x
Publication Date: July 5, 2017
Document code: bi_11.0.x_biolapserverag__en-us

Contents

Overview

The Administrator Manual is intended for administrators of large applications created with Infor BI OLAP Server. It covers such topics as setting up a client–server environment, network protocols and other technical information.

This handbook assumes that you are familiar with the client version of OLAP Server, in particular with browsing cubes and with logging on to OLAP Server.

Key components

To work with OLAP Server, it is important to have an overview of its three key components. These key components allow you to navigate through multidimensional cubes, select elements from dimensions and refer to cells in the multidimensional database from your spreadsheet.

The Cube Browser
This tool allows you to navigate through multidimensional cubes to arrive at the perspective of data you want to see. By rearranging buttons on a spreadsheet you can choose which dimensions of the multidimensional cube you want to see as row headings and which as column headings in your perspective. Using the Cube Browser you may also drill down through a dimension hierarchy.

The Dimension Browser
While the Cube Browser allows you to navigate through cubes, the Dimension Browser allows you to navigate through dimensions. You can select elements from a dimension and view the structure of a dimension (dimension hierarchy). It provides various means for moving through lists of elements, including drill-down and roll-up, search, and other features that allow you to create and look at different subsets of the elements within a dimension.

The DBGET Formula
This formula, and its siblings, is the key to linking spreadsheets with data in an OLAP Server cube. DBGET is a special spreadsheet function in OLAP Server. Its arguments are the cell address of the value you want to retrieve from a cube. The DBGET formula refers to a cell in an OLAP Server cube much the same way as a normal spreadsheet formula refers to another cell in the spreadsheet.

Standalone mode vs. client–server mode

There are two modules of software for OLAP Server: the Client and the Server. The client version of OLAP Server is also the stand-alone version of OLAP Server. It enables users to work independently of others on their local drive or on a laptop computer. It has the ability to load OLAP Server cubes into the memory of the end-user's computer

The client also allows users to access commonly shared database-server cubes and dimensions. By logging on to an OLAP Server, multiple clients can access and update the same data simultaneously.

The application server software allows various clients to shareOLAP Server data. The OLAP Server loads all cubes and dimensions specified by the administrator from disk into the memory of the OLAP Server computer. The data are no longer loaded into the memories of the computers that accesses the data. This insures that the data are only in one location. Since multiple computers are accessing data in the memory of one centralized computer, multiple users can now concurrently update one set of data.

Caution: It is important that you understand the difference between working locally and in client–server mode. In the former case you load OLAP Server databases into the memory of your computer. In the latter case you access data in the memory of a central OLAP Server computer.

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal.

If we update this document after the product release, we will post the new version on this website. We recommend that you check this website periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.

Installation

1

This chapter describes the hardware and software requirements and the installation of the OLAP Server.

Memory requirements

OLAP Server maintains all its databases in RAM. The total RAM necessary depends on the size of the OLAP Server application. If the entire application fits into RAM, performance is optimized. If swapping is necessary (that is, a virtual memory file is opened on disk), performance is reduced. The amount of RAM required by an OLAP Server application is approximately 1 MB.

Depending on the selected parameters (for example, cache size) the amount of RAM required may also increase. You can determine the estimated size of RAM required by a OLAP Server application by loading it either on a LOCAL OLAP Server (that is on the stand-alone version of OLAP Server) or on an OLAP Server.

Hardware requirements

The executable files (including help files etc.) of the OLAP Server require 70-100 MB of disk space. Additional disk space is required for the OLAP Server application (cubes and dimensions which will be loaded and made available by the server). If the OLAP Server application does not fit entirely into the RAM of the OLAP Server, additional disk space is required for the virtual memory "swap file ". Here again, the amount of space required depends primarily on the size of your OLAP Server application.

Note: The free disk space should be at least twice the size of the database. Insufficient disk space may result in data losses and server crashes.

Operating system requirements

The OLAP Server runs under Windows 7 or later. For additional requirements, such as a list of supported versions of used libraries, contact Support at Infor.

Note: Starting with OLAP Server 10.5.0 OLAP Server is no longer available for 32-bit operating systems. However, the 32-bit setup is delivered to enable installation of OLAP client core components.

Note: You can install this operating system as a workstation or a server, but the additional functionality of the server will not be utilized if installed as a workstation.

In all cases, the computer to be used as an OLAP Server must be installed on the Local or Wide Area Network as a standard network client. This client must have full read/write rights to the OLAP Server application directory.

Installation and location of files

This topic describes considerations that should be taken into account when selecting the location of the server executable and the application directory (that is, the directory containing the cubes and dimensions to be loaded by the OLAP Server):

- The server executable and the application directory need not be located on the hard disk of the computer that is used as an OLAP Server. However, having the application on the computer that is used as a server improves load performance.
- You may run up to 32 different applications on a single server computer. The applications, however, must all be loaded in different subdirectories under a single directory (subdirectory) on one hard-disk.
- The applications directory is best located on a file server, to which both the OLAP Server computer and the system administrator have access. This enables the network to include the cubes and dimensions in the application directory as part of the normal file-server backup routine.
- The OLAP Server manages access to the OLAP Server cubes and dimensions independent of the network access management. For clients that access cubes via the OLAP Server network access to the applications directory is not required. In fact, in the optimum configuration of the OLAP Server, clients should have no access to this network directory.

The cubes and dimensions used by the OLAP Server are normally stored in a file server directory called the Applications Directory.

Once loaded into memory, these dimensions and cubes are available for simultaneous access by clients. The OLAP Server also loads Control Cubes that restrict client access by cube and/or dimension elements. As the OLAP Server runs, it services requests from clients, sending data to spreadsheets and updating its cubes on request. When a client changes a shared value, the OLAP Server writes a record of that change to the file `Aleapr.txt` in the database directory. The records in this file comprise the transaction log. The transaction log serves both as an audit trail and as a means of restoring the database in case of hardware or power failure.

Numbering of versions for support purposes

All components that belong together share a unique identifying string to identify the version. For example, the string '10.5.0.506' contains these parts:

- 10.5.0: The version of the product that is sold.
- 506: This is the build number. All components that belong together have the same build number.
- Release: This parameter indicates whether this version was compiled in release or in debug mode. Debug versions are for the purpose of the Research and Development department only and are not distributed to customers.

Note: This parameter does not indicate the sales status of this product version. Even versions in interim, hotfix or beta phases have the string Release, because they are compiled in Release mode.

The build number is increased with every new version that is compiled. It is also increased if a component has not changed. The members of the Research and Development department may use the internal 4-section-number, for example, 11.0.0.183, to identify the changes in detail made between different versions of a component. This number has no implications for customer installations.

To identify the version of a particular OLAP Server component, use Windows Explorer® to locate the file. Right click the component, whose version you want to determine, and select **Properties** from the shortcut menu. A Windows dialog opens providing various tabs. Click the **Version** tab. You see a list box **Other version information** with a number of choices. Select **Product Version** to see the version number of the component.

See "File structure" on page 257 for a list of file names of all OLAP Server components.

File extensions

Three types of file are used by OLAP Server databases: system files, structure files, and non-critical files. OLAP Server system files are automatically generated the first time you log onto a database that you have created. They control security and access to the database. If they do not exist, users cannot log on to the database. OLAP Server system files all begin with the '#' symbol, for example: `#__dim__.dim`.

The second type of file are the database structure files, which are necessary for the OLAP Server database to contain data. These files are the dimension files and the cube files. Dimension files have a `*.dim` extension, cube files have a `*.ttt` extension. Subsets are stored in files with the extension `*.sst` or `*sst.xml`. Be careful when copying, moving or deleting these files: cubes are composed of dimensions, so moving or deleting a dimension used by a cube will render the cube unreadable. Deleting attribute files or subset files may prevent the dimensions that these files are associated with from opening correctly. As a rule, do not manually (that is, using Windows Explorer or DOS) rename or delete files within a database directory. The user interface of OLAP Server provides sufficient functionality to rename or delete files.

The third type of file are the non-critical files. These files may or may not exist. If one of these files is deleted, an error message may be displayed.

See "Installed files and their locations" on page 258.

This table lists the file extensions and gives a brief description:

File Extension	Description
`ttt`	Cube
`trs`	Cube transaction log
`dim`	Dimension
`drs`	Attribute transaction log
`sst`, `sst.xml`	Subset
`alv`	OLAP Server View
`alr`	OLAP Server Rules edited using Excel
`ald`	OLAP Server Dimension edited using Excel
`ini`	Configuration files
`txt`	Protocol files
`db_`, `rw`	Database administration files

Product features

2

This section describes the product features.

Infor BI OLAP Server Backup

The Backup functionality enables you to save an OLAP Server database to disk and copy it to a separate directory without having to shut down the server. You configure backups in Infor BI OLAP Administration. Backups can be scheduled or can be run by right-clicking a registered and connected database in OLAP Administration and selecting **Back up database**.

The backup does not include backing up the Infor BI Repository or any other files associated with the project.

Note: If defined, the `Db.ini` key `[Scheduler] BackupPath` overrides the path defined in the `Alea.ini` key `[Init] BackupRoot`.

This table describes the options to configure and schedule a backup. Select **Infor BI OLAP Administration > Favorite Databases > [database name] > Scheduler Settings**.

Options	Description
Server saving	
Scheduled backup of the server	Server backs up data at regular intervals.
Delete content of the backup directory	Deletes the content of the backup directory. **Caution:** Set to YES the content of the backup directory is deleted before backup.
First backup	Time of first back up. Apart from the time, you can also enter the value **Now** ; the backup will be carried out then some minutes after the start of the server.
Backup interval	Interval to back up.

Options	Description
Backup path	Path in the file system to save the backup.
Backup numbering	Set to YES, the suffix `nnnn` is attached to the backup path to keep multiple backup directories. `nnnn` is the number stored in Back up next number with leading zeros.
Back up next number	Next backup number. The server increases the backup number after every backup, no matter if the backup operation was scheduled or manually started.
Maximum backup number	If the backup number exceeds the specified maximum backup number, the Next backup number is set to 0. If -1 is set, the backup number is unlimited.

Performance counters for Infor BI OLAP Server and Event Agent

OLAP Server and Event Agent now support performance counters as known from Microsoft operating systems. Operators of OLAP Server are now enabled to log status and activity of OLAP Server databases using Microsoft Performance Monitor.

Note: If you intend to use performance counters of the Event Agent on Windows 7, the Event Agent must run with a full administrator access token. It might not even be enough if the current Windows user is in the Administrators group. Still applications normally run with only the level of access granted in a standard user access token. One way to grant full administrator access to the Event Agent is to set the database startup mode to `"Automatic + Events"` and then start the Communication Manager with administrative privileges.

Start Microsoft Performance Monitor via **Start > Control Panel**. Then double-click **Administrative Tools** and double-click **Performance**.

For information about creating and configuring performance counters refer to the Microsoft online help.

Available counters for Infor BI OLAP Server

Performance object	Counter	Description
Event Agent	AttributeChange Events	Number of recorded and forwarded AttributeChange events

Performance object	Counter	Description
	CellChange Events	Number of recorded and forwarded CellChange events
	DimensionChange Events	Number of recorded and forwarded DimensionChange events
	Event Agent Commands	Number of recorded Event Agent commands
	Failed AttributeChange Events	Number of failed AttributeChange events
	Failed CellChange Events	Number of failed CellChange events
	Failed DimensionChange Events	Number of failed DimensionChange events
	Failed Generic Events	Number of failed Generic events
	Failed ProtocolError Events	Number of failed ProtocolError events
	Generic Events	Number of recorded and forwarded Generic events
	ProtocolError Events	Number of recorded and forwarded ProtocolError events
	Total Event Filters	Number of event filters
	Total Events	Number of all recorded events (attribute changes, cell changes, dimension changes, protocol errors, generic events, Event Agent commands)
	Total Events triggered	Number of all recorded and forwarded events (attribute changes, cell changes, dimension changes, protocol errors, generic events, Event Agent commands)
	Total Events/sec.	Number of all recorded events per second (attribute changes, cell changes, dimension changes, protocol errors, generic events, Event Agent commands)
OLAP Server: Connection	Active Users	Number of active users
	Current Connections	Number of connections to the OLAP Server

Performance object	Counter	Description
OLAP Server: Events	Event Subscribers	Number of clients receiving events from OLAP Server
	Events Waiting	Number of events in the OLAP Server queue
	Total Events	Number of events processed since the start of the OLAP Server
OLAP Server: General	Base Values	Number of basic values in the OLAP Server database
	Cached Values	Number of cached values in the OLAP Server database
	Changed Values	Number of values inserted/changed since the start of the OLAP Server
	Maximum Rule Recursion	Deepest recursion for rules calculation
	Threads	Number of threads in the OLAP Server
	Used CSD Slots	Number of CSD slots used by the OLAP Server kernel
OLAP Server: Startup	Server Uptime	Time elapsed since the start of the OLAP Server
	Total Dimensions	Number of dimensions stored in theOLAP Server database
	Total Elements	Number of elements stored in the OLAP Server database
	Total Hypercubes	Number of cubes stored in the OLAP Server database

Export AleaLog to relational database

AleaLog

OLAP Server can log every cell change made by write and delete operations to a log file. One example of a write operation is splashing. After those operations, the server writes one or more entries to the log file, with the information about the operation, among other things such as the value. In some cases, the values are not logged as delivered but as the computed end value. For example, if you have the value 1 in a specific cell and the delta operation applies a delta value of 2, then the server writes 3 as the value.

This file includes information about the date, time, user name, cube name, cell address, original value, and new value. The file is created in the database directory specified in the initialization files. It is named `Alealog.txt` and it is tab-separated. This file can be used, if a database needs to be recreated, or user activities need to be monitored. It needs to be maintained manually, otherwise it becomes larger and larger.

Unlike the transaction log, this file needs to be maintained manually. It will get larger and larger and should be edited periodically. There are three options that determine which operations are to be logged:

- ALL: Log all client operations
- TRANS: Log only those operations that are also logged in the transaction log
- NONE: No operations are logged

1. Select the Log Settings component.
2. Select **All** or **Trans** from the **Definition of the cubes** list.

Export AleaLog to relational database

This feature enables OLAP Server to log changes of data cells into a relational database for persistent storage.

Cell values changes are not logged if they get lost because of these events:

- The deletion of the entire cube
- The deletion of a dimension element the cell belongs to
- The change of cell type from S to N or N to C

These changes are not logged:

- Attributes
- Dimensions
- Cell Comments
- Cubes structures, for example, changing the number of dimensions
- Configuration settings

Note: If logging to SQL is enabled, everything will be logged, the settings for `Alealog.txt` are ignored. If it should be done partially only, the filtering has to be done inside the SQL Server.

Scheduler

The new task is created in the OLAP Server scheduler (`Db.ini\SCHEDULER\CELLTOSQL`).

It is executed every 10 seconds: It locks the `Alealog.txt` file mutex, reads its content, writes it to the database, empties `Alealog.txt` and unlocks it.

Note: `Alealog.txt` is only emptied if the insertion to the database succeeds.

Caution: Splashing information in `Alealog.txt` will not be written into the relational database.

Turning this scheduler on internally `LoggedOperations=ALL` (the local `Db.ini` file is not changed).

Setting up the feature

Note: We recommend that you delete the `Alealog.txt` file before activating the Export AleaLog to Relational Database feature.

The regional settings of the OLAP Server must be compatible with the regional settings of the relational database.

If there is no table with the name defined in the `CellToSQLTable` key in the relational database, the OLAP Server creates one.

In order to use Export AleaLog to Relational Database with 30 dimensions you must use LogVersion=4.

- Create a database in the relational database (for example, Alealog)
- Change these keys in the `Db.ini` file:

```
[SCHEDULER]
CellToSQL=YES
CellToSQLConnectionString='Server=localhost;Trusted_Connection=yes;
Database=Alealog;'
CellToSQLTable=Alealog
```

Caution: Splashing information in `Alealog.txt` will not be written into the relational database.

Connection Strings

Microsoft SQL Server

Local SQL Server. For example, OLAP Server and SQL Server are running on the same computer:

```
DRIVER={SQL Server}; SERVER=(local); Trusted_Connection=yes;
DATABASE=Alealog;
```

Remote SQL Server. For example, OLAP Server and SQL Server are running on different computers:

Windows authentication:

```
DRIVER={SQL Server}; Server=<remote_computer_name>;
Trusted_Connection=yes;Database=Alealog;
```

In this case there must be a Windows authentication logon within the remote SQL server for user credentials under which the OLAP Server is running.

SQL authentication:

```
DRIVER={SQL Server}; Server=<remote_computer_name>;
Trusted_Connection=no;Database=Alealog;
UID=<user_sql_authentication>;PWD=<correct_password>;
```

Oracle

```
DRIVER={Oracle in OraClient11g_home1}; DBQ=<connection alias>;
UID=<user>; PWD=<password>;
```

Note: The default for the service name is ORCL , but for DBQ the Connection Alias needs to be entered.

MySql

```
DRIVER={MySQL ODBC 3.51 Driver}; SERVER=localhost; DATABASE=alealog;
UID=<user>; PWD=<password>;
```

Postgres

```
DRIVER={PostgreSQL ODBC Driver(Unicode)}; SERVER=localhost; PORT=5432;
 DATABASE=AlealogPG; UID=myuser; PWD=mypw; OPTION=A6=;B1=1000001;
```

Affected .ini keys

These keys are affected:

- CellToSQL
- CellToSQLConnectionString
- CellToSQLTable

- SQL: Overwrite definition of the cubes
- LogVersion

See "Db.ini" on page 271

Version handling

`Alealog.txt`

The hierarchy names are stored in separate fields, each directly before the dimension element name. `AleaLogVersion.txt` shows which version has been used on writing the `Alealog.txt` file. This version file is checked on server start and on load config.

If the version stored in `AleaLogVersion.txt` is different from the version in `Db.ini` and there are records in `Alealog.txt`, then the log file is preserved as `AleaLog_Vx.txt`, where `x` is the version.

Relational tables

To use AleaLog to Relational Database LogVersion=5 with additional columns for hierarchy names must be used. There is a compatibility mode for older versions with concatenation (`<hierarchy name>[<element name>`) with character '`[`' as separator.

The default hierarchy is logged with hierarchy name.

The structure of the relational tables are checked at server start The length of the columns must be equal or greater than the required column length (if hierarchy names and elements names are concatenated in older log versions).

Cache partitioning

The OLAP Server saves calculated values in the cache to retrieve them quickly instead of recalculating them. Changed values and dependent cells are invalidated (deleted from the cache) according to the dependencies in the cache partitions. Due to these saved dependencies between cache partitions, it is not necessary to delete the whole cube cache, but only dependent partitions. This is also valid for dependencies between different cubes.

As the number of partitions increases, the value of adding addition partitions rapidly diminishes. For example, adding partition 101 to 100 partition would only change the average amount of data by 1/10,000 the of the total cube size. However, the overhead per partition is constant. Therefore it is recommended to use this feature with a small dimension. One partition is created for each element of the dimensions. The maximum number of elements is 256. If the dimension contains more than 256 elements, partitioning is disabled and a message is written into the `Aleapr.txt` log file.

Example: Enabling Cache Partitioning

Enable Calculated values cache by entering a value greater than 0 in OLAP Administration.

To indicate the dimension "YEARS" as to be partitioned, select YEARS from the **Cache-Partition** list on the Properties dialog box of a cube:

```
<Alea:Properties xmlns:Alea="http://www.misag.com">
<Alea:CachePartitionDimension Name="YEARS"/>
</Alea:Properties>
```

For some use cases more than one cube must be properly partitioned.

If Calculated values cache is disabled, the Cache Partitioning setting has no effect.

Example: Partitioning on the Version dimension of a planning cube

You have a cube with two versions, actual and budget. The data for the actual version is imported once in a budget cycle and is read-only. The aggregated values are only calculated once. The budget data is constantly updated as it is entered by users. If there is no partition, the aggregated actual data has to be recalculated each time budget data is entered. Partitioning the cache on the version dimension allows the cached actual data to be maintained while the budgeting data changes.

Handling of certificates for the HTTPS connection

Parameter	Required format	Default value	Description
Alea.ini [HTTP]			
RootCertificate	File name	`root.pem`	Truststore, certificate file in pem format
Certificate	File name	`server.pem`	Certificate file in pem format containing the server certificate and the associated private key. The private key can be stored encrypted.

The OLAP Server allows to use for the SSL based communication either the certificates generated at setup time (these are self-signed), or certificates created by the customer/vendor.

To use other/your certificates for the OLAP Server, replace the files `olap_root.pem` and `olap_server.pem` with appropriate certificates you have generated:

`C:\ProgramData\Infor\BI\OLAP\certificates\olap_root.pem`

`C:\ProgramData\Infor\BI\OLAP\certificates\olap_server.pem`

You can change the location and name of those files with a setting in the HTTP section of the `Alea.ini` file:

```
Alea.ini [HTTP] RootCertificate=%YourPath%\%YourFileName%
Alea.ini [HTTP] Certificate=%YourPath%\%YourFileName%
```

Generating an InforCA-signed certificate

For SSL-based communication you can use the certificates that are generated at setup or you can generate your own InforCA-signed certificates.

To generate a certificate:

1. Install the latest version of Win32OpenSSL-0_9_8r.
2. Open `openssl.cfg` in a text editor.

   The file is in the `bin` folder of the OpenSSL installation.
3. Edit these parameters of `openssl.cfg` as required:

   countryName_default
   Specify the two-digit ISO code of the country in which the server is located. .

   stateOrProvinceName_default
   Specify the name of the state or province name in which the server is located.

   localityName_default
   Specify the city in which the server is located.

   0.organizationName_default
   Specify the name of the organization that controls the server.

   1.organizationName_default
   Optionally, specify the name of a second organization. For example, specify the name of a subsidiary of the parent organization.

   organizationalUnitName_default
   Optionally specify a business unit or division within the second organization.
4. Run `cmd.exe` as Administrator.
5. Navigate to the `bin` directory of the OpenSSL installation.
6. Use this syntax to create a host private key to PKCS#10 standard:

   ```
   openssl req -newkey rsa:1024 -keyout host_[server_name].key -nodes -config openssl.cfg -out host_[server_name].req
   ```
7. When prompted to enter information to incorporate in your certificate request, specify the parameter values that you entered into openssl.cfg. When prompted, specify these values:

   Common Name
   Specify the hostname or fully qualified domain name of the OLAP Server.

Note: If you specify the fully qualified domain name of the server, you must also specify **YES** as the value of the `Fully qualified domain name` flag in OLAP Administration. Select **Local computer > Communication**. The `Fully qualified domain name` flag is in the General section.

Email Address
For example: `noreply@example.com`

8 Browse to https://certificate.infor.com/certsrv/.
9 Click **Request a Certificate**.
10 Click **Advanced Certificate Request**.
11 Copy and paste the contents of `host_[server_name].req` into the **Base-64-encoded certificate request (CMC or PKCS #10 or PKCS #7):** field.
12 Select **Web Server** in the **Certificate Template:** field.
13 Click **Submit**.
14 On the Certificate Issued page, select **Base64 encoded** and click **Download certificate**.
The certificate is downloaded as `certnew.cer`.
15 On the Certificate Issued page, select **Base64 encoded** and click **Download certificate chain**.
The certificate chain is downloaded as `certnew.p7b`.
16 Rename `certnew.cer` to `host_[server_name].cer` and rename `certnew.p7b` to `Infor IssuingCA.p7b`.
17 Move `certnew.cer` and `InforIssuingCA.p7b` to the bin folder of the OpenSSL installation.
18 In the Command Prompt, specify this command to concatenate key and certificate: `copy host_[server_name].cer + host_[server_name].key host_[server_name].pem`
19 In the Command Prompt, specify this command to split the certificate chain according to PKCS#7: `openssl pkcs7 -in InforIssuingCA.p7b -out InforIssuingCA.pem -print_certs`
20 Stop the Infor BI OLAP Service Manager service.
21 Open Alea.ini and specify these paths in the [HTTP] section:
```
RootCertificate=<complete_path>\InforIssuingCA.pem
Certificate=<complete_path>\host_[server_name].pem
```
22 Test the certificate by browsing to `https://[server_name]:8211`.
If the certificate is configured correctly, the Infor BI OLAP Server Development page is displayed.

Inaccessible members

The implementation of the MDX Engine of OLAP Server before version 10.3 was created with the base rule to be consistent. This means if a user had no permission to see an element, an MDX statement using that member unique name resulted in an error.

Examples:

- The creator of a report defines a number of members on an axis, and the system shall display the subset of the elements that the user has access to.
- The creator of a report wants to show the children of a member that are visible to the user. But the starting member is not visible to the user.

The user may use different kinds of members in an MDX statement:

- Visible - user has at least read permission
- Inaccessible - members are not visible to the user due to permission restrictions. If he would know that they are there, even though he is unable to see the values, would be information.
- Not existing at all - these members do not exist (for example, the user has mistyped the name)
- Not existing anymore - these members have been renamed or removed

Starting with OLAP Server 10.3 the MDX Engine allows the user to use inaccessible members in an MDX statement (for example, using the functions CHILDREN and DESCENDANTS): the operation is not stopped with an error message, but it is finished and returns only the visible members. Invisible members are never part of the XMLA response.

Allow Access to Inaccessible Members (Database setting)

Parameter	Required format	Acceptable values	Default value	Description
`AllowReferenceToInaccessibleMembers`	Value	YES, NO	NO	Allow reference to inacessible members. If set to Yes, MDX statements can contain references to inaccessible members.

The setting is stored in the `INIT` section of the `Db.ini` and is changeable through the OLAP Administration (not during runtime).

As the default value for `AllowReferenceToInaccessibleMembers` is set to `NO` , it will have no influence on existing systems.

Enable Access to Inaccessible Members (Session setting)

The setting is valid for a session, it can be changed by each user or - more precisely - by his client for his own session. There is no special permission needed in order to change the setting. While the user is connected, the configuration is available. As soon as the session is ended, the configuration is discarded. If there are multiple sessions for the same user, the configuration is independent for the different sessions. The configuration is changeable during the existence of the session.

Name: `EnableAccessToInaccessibleMembers`

Possible values: TRUE | FALSE

Default value: TRUE

Note: As long as the database setting `AllowReferenceToInaccessibleMembers` is set to NO, it is without effect

Infor BI Application Studio and Infor BI Office Plus

Functions returning single members will return #NA if the member does not exist, value and property requests for invalid members will return #NA.

Properties

OLAP Server supports defining properties for database, connections, cube and dimensions in XML format.

Database

Define the properties of a database using the XML function Database `PutProperties` or in the Extended Properties of a server in the OLAP Administration.

Tag	Attribute	Format	Value	Description	Action on database	User rights
Alea:Database	Name	Text	Database name		Read	
Alea:Path		Text	Database path		Read	
Alea:ReservationMode		Bool	True | False (Case Sensitive)		Read	
Alea:Users	Connected	Number	Currently attached user		Read	
Alea:EventAgent	Status	Enum	Unknown | Running | Stopped | Paused | NotRunning		Read	
Alea:EventAgent	SessionID	Text	ProcessIdToSessionId (Windows)		Read	
Alea:Configuration	DesignerExclusive	Bool	true | false	Block Server to all users except Designer	Write Read	Administrator Everybody

Connection

Define the properties of a connection using the XML function `Connection PutProperties` or in the Extended Connection Properties of a server in the OLAP Administration.

Tag	Attribute	Format	Value	Description	Action on connection	User rights
Alea:Splashing						
Alea:UndoRefNumber		Number	Number of Undofile		Read	
Alea:Events	Generate	Bool	true | false	Set or remove the Generate Events flag	Read, Write	
Alea:Translation	LocaleIdentifier	Number	Language ID (default: ID=0)	Set the ID of the prefered language.	Write	

Cube

Define the properties of a cube using the XML function `Cube PutProperties` or in the Extended Properties of a cube in the OLAP Administration.

Tag	Attribute	Format	Value	Description	Action on cube	User rights
Alea:Rules	Number	Number	Rule count for user interface (number of enabled <Rule> XML nodes)		Read	
Alea:MeasureDimension	Name	Text	Dimension name	Specify the Measure dimension	Read, Write	Administrator
Alea:MeasureDimension	ExplicitlyDefined	Bool	true | false	One ID for the Measure dimension	Read	
Alea:TRS	Enabled	Bool	true | false	TRS file is generated for the cube	Read, Write	Administrator

Tag	Attribute	Format	Value	Description	Action on cube	User rights
Alea:Error-Suppression	Enabled	Bool	true | false	Do not omit an error when C cells are modified	Read, Write	Administrator
Alea:Access-Cube	Enabled	Bool	true | false	Enable Access control	Read, Write	Administrator
Alea:CachePartitionDimension	Name	Text		Specify the name of the cache partition dimension	Read, Write	Administrator
Alea:MDACList				Specify one or more MDACs by name		
Alea:MDAC	Name	Text	MDAC name	Specify the MDAC by the cube name	Read, Write	Administrator
Alea:Translation	LocaleIdentifier	Number	Language ID		Read, Write	Administrator
Alea:Caption		Text	Cube caption		Read, Write	Administrator
Alea:Description		Text	Dimension description		Read, Write	Administrator

Dimension

Define the properties of a dimension using the XML function `Dimension PutProperties` or in the Extended Properties of a server in the OLAP Administration.

Tag	Attribute	Format	Value	Description	Action on dimension	User rights
Alea:ODBO-Type		Number	0 (unknown) 1 (Time) 2 (Measure) 3 (Other) 5 (Quantitative) 6 (Accounts)		Read	Administrator / Change Dimension

Tag	Attribute	Format	Value	Description	Action on dimension	User rights
			7 (Customers) 8 (Products) 9 (Scenario) 10 (Utility) 11 (Currency) 12 (Rates) 13 (Channel) 14 (Promotion) 15 (Organization) 16 (Bill of Materials) 17 (Geography)			
Alea:DefaultMember	Name	Text	Default Element	Default element for ODBOType	Read, Write	Administrator / Change Dimension
Alea:AccessCube	Name	Text	Physical name of DAT table	Access cube for the dimension	Read, Write	Administrator
Alea:MemberKey	AttribTableID	Text	Numeric Value (1-3)	ID of the dimension attribute table	Write	Administrator / Change Dimension
Alea:MemberKey	FieldName	Text	String Field Name	Name of a field in the attribute table	Write	Administrator / Change Dimension
Alea:Hierarchy	Reverse	Bool	true | false		Read	Administrator / Change Dimension
Alea:Translation	LocaleIdentifier	Number	Language ID		Read, Write	Administrator / Change Dimension
--Alea:Caption		Text	Dimension caption		Read, Write	Administrator / Change Dimension

Tag	Attribute	Format	Value	Description	Action on dimension	User rights
--Alea:Description		Text	Dimension description		Read, Write	Administrator / Change Dimension
--Alea:MemberCaption	AttribTableID	Number			Write	Administrator / Change Dimension
--Alea:MemberCaption	FieldName	Text			Write	Administrator / Change Dimension
--Alea:LevelCaption		Text			Write	Administrator / Change Dimension
--Alea:LevelCaption	Level	Number	0..n	Level caption	Write	Administrator / Change Dimension

Multilingual captions and descriptions

OLAP Server supports multilingual captions for cubes, dimensions, levels and members and multilingual descriptions for cubes and dimensions. The translations are stored as system properties of cubes and dimensions, current cube and dimension descriptions are moved to translations for neutral language.

Define the properties using the XML function `Dimension PutProperties` or in the Extended Properties of a server in OLAP Administration.

Specifying the preferred language

The client application can specify its preferred language in the connection property.

```
<Alea:Properties xmlns:Alea="http://www.misag.com">
   <Alea:Translation LocaleIdentifier="1031"/>
</Alea:Properties>
```

The locale identifier (LCID) XML-attribute is a decimal number (for example 1031 German [Germany], 1033 English [United States], 0 Neutral local language). The XMLA specification uses LCID as a locale identifier property as well.

Each language identifier is composed of a primary language identifier indicating the language and a sub language identifier indicating the country/region. The language identifier corresponds to a particular locale, for example, English (United States), represented as "en-US". The language identifier is used as part of the locale identifier.

See http://msdn.microsoft.com/en-us/library/ms776294.aspx

Required OLAP permissions

Permission	Description
View	Users with View permission can read all translations
Administer OLAP Database	Users with Administer OLAP Database permission are allowed to create, change or remove cube translations
Edit Dimensions	Users with Edit Dimensions permission are allowed to create, change or remove dimension, level and member translations

Language order

Each language has its locale identifier and primary language identifier. For example Austria (AT) has the LCID 3079 (hexa c07) and primary language identifier 7.

It is also possible to define translations for the neutral language in OLAP Server. This language has identifier 0. If an object (such as cube, dim) has captions for more languages the first non-empty caption is taken.

The order of languages where captions are searched is defined in this way:

1. Language defined by the Locale identifier
2. Language defined by the Primary language identifier
3. Neutral language
4. Object name (for example, level name for level caption)

The order of languages where descriptions are searched is:

1. Language defined by the Locale identifier
2. Language defined by the Primary language identifier
3. Neutral language

Example:

The client application is connected with LCID=3079 (German - Austria).

Element 'Jan' has captions defined for LCID=3079 'Jänner', for LCID=7 'Januar' and for the neutral language (LCID=0) 'January'

Element 'Feb' has captions defined for LCID=7 'Februar' and for the neutral language 'February'

Element 'Nov' has a caption defined for the neutral language 'November'

Element 'Dec' has not captions defined.

This can be defined in a table, too:

Member	**Captions**		
Name	**Neutral**	**German**	**German-Austria**
Jan	January	Januar	Jänner
Feb	February	Februar	
Nov	November		
Dec			

Then caption of:

'Jan' is 'Jänner' (taken from the attribute German-Austria)

'Feb' is 'Februar' (taken from the column German, as the column German-Austria is not filled)

'Nov' is 'November' (taken from the column Neutral as no German string is available)

'Dec' is 'Dec' (member name, as no caption is available)

Add or change translations

Cube translations are added by a an XML request. If a translation for a language already exists it is completely removed before the new translation is added.

```
<Alea:Properties xmlns:Alea="http://www.misag.com">
   <Alea:Translation LocaleIdentifier ="3079">
      <Alea:Description>Austrian description</Alea:Description>
      <Alea:Caption>Austrian caption</Alea:Caption>
   </Alea:Translation>
   <Alea:Translation LocaleIdentifier ="7">
      <Alea:Description>German description</Alea:Description>
   </Alea:Translation>
</Alea:Properties>
```

The XML request for dimension translations is very similar. Additionally it can contain level and member captions.

```
<Alea:Properties xmlns:Alea="http://www.misag.com">
   <Alea:Translation LocaleIdentifier="3079">
      <Alea:Caption>Monat - Caption</Alea:Caption>
     <Alea:Description>Österreichische Monatsnamen</Alea:Description>

      <Alea:LevelCaption Level="0">Jahr</Alea:LevelCaption>
      <Alea:LevelCaption Level="1">Quartal</Alea:LevelCaption>
      <Alea:LevelCaption Level="2">Monat</Alea:LevelCaption>
      <Alea:MemberCaption AttribTableID="1" FieldName="CaptionAT"/>
      </Alea:Translation>
   <Alea:Translation LocaleIdentifier="7">
      <Alea:Description>German description</Alea:Description>
   </Alea:Translation>
</Alea:Properties>
```

Remove translations

Translations can be removed by sending a XML request containing an empty translation tag.

```
<Alea:Properties xmlns:Alea="http://www.misag.com">
   <Alea:Translation LocaleIdentifier="1029"/>
</Alea:Properties>
```

OLAP Content Packer

The OLAP Content Packer allows to pack OLAP databases into a content package.

To create a package, specify this information:

- Where the model is located on the hard drive
- The name of the model in the farm, when it is deployed
- The name of the resulting package file, with the extension `.app_olap`.

Example:

You want to create a content package with these characteristics:

- Data model: Best Practices
- Location: `C:\Olap_Models\Best_Practices`
- Name of the model in the farm: BestPracticesModel
- Name of the content package: `Best_Practices_Data.app_olap`

Corresponding call:

```
Infor.BI.Olap.ContentPacker.exe C:\OLAP_Models\Best_Practices
BestPracticesModel Best_Practices_Data.app_olap
```

Example:

It is possible to transport some configuration data alongside the data model, which is set in the farm when the OLAP data model is deployed. To add configuration information to the `Db.ini` file, you need to pass its path in the command:

```
Infor.BI.Olap.ContentPacker.exe C:\OLAP_Models\Best_Practices
BestPracticesModel Best_Practices_Data.app_olap
C:\OLAP_Models\Best_Practices\Db.ini
```

The global `Db.ini` file is not taken in account during the creation of the package. Only the settings in the `Db.ini` file specified in the command are considered.

This means you must set the Save all parameters in the `Db.ini` file option to Yes in OLAP Administration before creating the content package.

Dynamic attributes

3

Dynamic attributes are a way to store dimension element attributes that may depend on the context of a cell. The value of an attribute therefore depends on some elements of other dimensions in the cube. Dynamic attributes can be used in scenarios where attribute values change depending on the selected elements in other dimensions.

For example, a product dimension with a responsible product manager assigned to each product or product group. This information changes over time. Using the time dimension as a driver dimension, it is possible to store the responsible product managers for every point a time. In this case, it is necessary to store the attributes on aggregated cells as well.

Terminology

Term	Description
Dimension Attributes	The known attributes defined on an OLAP dimension. They can be filled for all elements of a dimension.
Dynamic Attributes	These are the attributes that may depend on the context of a cell.
Dimension with dynamic attributes	Dynamic attributes are defined in the context of a dimension with dynamic attributes.
Driver dimensions	Elements in these dimensions drive the context in which a dynamic attribute is resolved.
Fact cube	The cube which holds the actual fact data. Dynamic attributes are used in the context of the cube.
Dynamic attribute cubes	A dynamic attribute cube stores the dynamic attributes for a dimension with dynamic attributes.
Attribute dimension	The attribute dimension holds the list of attributes.

Functionality

The fact data is stored in the fact cube. This cube contains dimensions which can be categorized as dimensions with dynamic attributes, driver dimensions and other dimensions.

The dimensions with dynamic attributes are the dimensions where the dynamic attributes should be available on the elements. For dimension attributes it would be enough to know the element name and the attribute name in order to read the attribute value for that element. In the case of a dimension with dynamic attributes the value of the attribute is valid only in the context of a cell. But it does not depend on all dimensions of the cube, but just on the driver dimensions or a subset hereof.

Example:

There is a dimension Employees in the cube and the attribute should be Bonus in %. But in different scenarios (elements of the scenario dimension), an employee has different bonus rates. Therefore the bonus rates cannot be stored in the dimension attributes.

Scenario	Optimistic			
Measure	Base Salary			
		Months		
Employee	Bonus in %	January	February	March
IT		8.300	8.300	8.300
Claus Mayer	20%	3.000	3.000	3.000
Reinhold Ebermann	10%	2.500	2.500	2.500
Thomas Mustermann	40%	2.800	2.800	2.800

Scenario	Pessimistic			
Measure	Base Salary			
		Months		
Employee	Bonus in %	January	February	March
IT		8.300	8.300	8.300
Claus Mayer	10%	3.000	3.000	3.000
Reinhold Ebermann	2%	2.500	2.500	2.500
Thomas Mustermann	25%	2.800	2.800	2.800

Depending on the selection of an element in the scenario dimension, the values of the dynamic attribute changes.

Limitation: The dynamic attribute Bonus in % can only be returned if the driver dimension scenario is in the slicer of the request.

Months	January		
Measure	Base Salary		
		Scenario	
Employee	Bonus in %	Optimistic	Pessimistic
IT		8.300	8.300
Claus Mayer		3.000	3.000
Reinhold Ebermann		2.500	2.500
Thomas Mustermann		2.800	2.800

In this example the scenario dimension is in the column header. Therefore it is not possible to show a value for the attribute, as different ones are defined for the different scenarios.

It is possible to use more than one driver dimension in a cube supporting dynamic attributes. But not all driver dimensions affect each dynamic attribute. Therefore the information must be stored on which driver dimension an attribute depends.

For the calculation of the dimension properties inside an MDX statement it is important to know if the dimension is a dimension with dynamic attributes. In that case the driver dimensions must be identified as well. This identification of the driver dimensions is stored with the cube in the extended cube properties.

The dynamic attribute data is stored in a dynamic attribute cube. Each dimension with dynamic attributes stores its dynamic attributes in a dedicated cube. The name of the cube is stored in the extended properties of the fact cube. The dynamic attribute cube has these dimensions:

- The driver dimensions
- The dimension with the dynamic attributes
- The attribute dimension

The attribute dimension is the measure dimension in the cube.

The attribute dimension contains a list of all dynamic attributes of the attribute dimension with dynamic attributes. Additionally it contains these dimension attributes:

- For each of the driver dimensions there is a Boolean attribute that defines if this dynamic attribute depends on the driver dimension. True means, it depends on the driver dimension. The name of the attribute is defined in the extended properties of the dynamic attribute cube.
- A Format_Type attribute holds format information of the attribute.

Depending on the type of data to be stored, the elements in the attribute dimension is either of type N (numeric) or S (string).

Dynamic attributes can be updated by writing to the attribute cube directly. The OLAP Server ensures that an update into the cells of that cube has the same effect on the time stamps as an update to the dimension properties. It is not be possible to update the dynamic attributes using attribute write requests in the fact cube.

Schema and MDX

The schema MDSCHEMA_PROPERTIES has been extended to list the dynamic attributes. A new value (16) in the column PROPERTY_TYPE marks a dynamic property.

The schema MDSCHEMA_DIMENSIONS has been extended by these columns:

- DYNAMIC_ATTRIBUTE_TYPE

  This gives information if the dimension is connected to dynamic attributes. These are the possible values:

  - Dimension with dynamic attributes (1)
  - Driver dimension for at least one driver dimension (2)
  - Other dimension (0) - default

- DYNAMIC_ATTRIBUTE_CUBE

  The name of the cube that holds the dynamic attribute values for a dimension with dynamic attributes

The dimension properties in MDX have been extended so that dynamic attributes can be returned as dimension properties. This works only in the case that the driver dimensions are in the slicer of the calculation, hence in the where clause. If the dimension is not mentioned at all in the MDX statement this is fine, as in that case the default element specifies the context on that dimension.

The schema MDSCHEMA_CUBES has been changed. The column CUBE_TYPE currently holds information like TAC, DAC, MDAC and USER. This has been extended by the type ATTRIBUTE to mark attribute cubes.

Data storage

The extended properties of the fact cube store the information which dimensions support dynamic attributes:

```
<Alea:DynamicAttributes>
   <Alea:Dimension Name="ATD_Employees">
      <Alea:AttributeCube Name="ATCData_Employees" />
   </Alea:Dimension>
      <Alea:Dimension Name="ATD_Jobs">
      <Alea:AttributeCube Name="ATCData_Jobs" />
   </Alea:Dimension>
   …
</Alea:DynamicAttributes>
```

This information is stored:

- Dimension: This tag is repeated for each dimension that supports dynamic attributes. The name of the dimension is stored here.
- AttributeCube: The name of the cube that contains the attribute data.

On the dynamic attribute cube additional information is stored.

```
<Alea:DynamicAttributeConfiguration>
   <Alea:Dimension Name="ATD_Employee" />
   <Alea:AttributeDimension Name="ATDAttribute_Employees" />
   <Alea:DriverDimensions>
   <Alea:DriverDimension Name="ATD_Cycles">
      <Alea:ApplicabilityAttribute Name="Cycle_applicable" />
      <Alea:NonApplicableElement Name="N.A." />
   </Alea:DriverDimension>
   <Alea:DriverDimension Name="ATD_Entities">
      <Alea:ApplicabilityAttribute Name="Entity_applicable" />
      <Alea:NonApplicableElement Name="N.A." />
   </Alea:DriverDimension>
   …
   </Alea:DriverDimensions>
</Alea:DynamicAttributeConfiguration>
```

This information is stored:

- Dimension: The name of the dimension that is receiving the dynamic attributes
- AttributeDimension: The name of the dimension holding the attributes
- For each driver dimension:
- DriverDimension: The name of the driver dimension
- ApplicabilityAttribute: The name of the attribute that defines if the driver dimension is applicable for this attribute. The default is true.
- NonApplicableElement: The name of the element that is used in this dimension if the driver dimension is not applicable.

Hierarchies

4

Hierarchies in cubes and dimensions

All base elements are available to every hierarchies. Each base element must be a part of at least one hierarchy. Base elements with the same name but in different hierarchies are the same element and all data are shared. This includes cell value, cell note and attributes.

The union of all base elements of all hierarchies form the base elements of the dimension.

Unlike base elements, aggregated elements only exist in the hierarchy they are defined in. Separate hierarchies can contain aggregated elements with the same name. These elements do not share attribute values or point to the same cube data.

Aggregated elements with the same name that appear in different hierarchies are treated as different elements by the database kernel. As a result, the values calculated for such elements can vary between hierarchies.

As described above, it is possible to create in multiple hierarchies aggregated elements with the same name. But the same name cannot be used for a base element and an aggregated element. This is even not allowed if the base element and the aggregated element are located in different hierarchies.

Each dimension has at least one hierarchy. There is a default hierarchy on each dimension. In case that a dimension has multiple hierarchies and none of them was selected as the default hierarchy, OLAP Server specifies one.

If no hierarchy has been explicitly created, OLAP Server creates an implicit hierarchy. The name of this hierarchy is the same as the name of the dimension.

Each hierarchy has a default member. If not explicitly selected this is one of the top level members after the permissions have been applied.

There are two kinds of hierarchies in OLAP Server dimensions: parent child hierarchies and attribute driven hierarchies. Internally, these two types of hierarchies are identical. The only difference between them is the method used to define them. Parent-child hierarchies are defined by explicitly specifying each parent child relationship in the hierarchy. Attribute driven hierarchies are defined by treating one or more attributes of the base elements of the hierarchy as an aggregated level. OLAP Server automatically generates the parent-child relations for the user.

Attribute hierarchies are not updated immediately after an attribute changes or an element has been added to the source hierarchy. This only happens if the method `Dimension.UpdateHierarchies` is called, or if the hierarchy definition changes.

Hierarchies and rules

In version 10.6 and earlier, elements can be addressed in rules in two different situations, either in static references like `[Units, 'TIME':'January']` or in functions like `DB('Totsales','2015', 'Actual',…)`. The important difference is that the parameters of the function are explicitly dedicated to a certain dimension. For example the parameter '2005' is resolved in the first dimension of the cube 'Totsales' only. If it is not found there, an error is returned.

The static references are more flexible. In the example `[Units, 'TIME':'January']` it is not clear to which dimension the element Units belongs to. Therefore the OLAP Server tries to resolve it in each dimension of the cube. If the element name is unique, the rule is valid. If the element name is not unique it has to be preceded by the dimension name otherwise the rule is invalid. 'Time':'January' is an example for the correct combination. The term can be enclosed by single or double quotes to clearly define the beginning and the end of the string.

With the introduction of the hierarchies an element in a static reference can be addressed either by

- The element name - as in earlier versions.
- The dimension name, the colon and the element name - as in earlier versions.
- The dimension name, the colon, the hierarchy name, the colon and the element name.

Other combinations, especially hierarchy name, colon, element name are not be supported.

There are rule functions that accept element names as parameters and have an implicit dimension reference. In order to be able to pass hierarchies in there as well, the function `ElementInHierarchy` has been implemented. It takes two parameters to clearly identify an element, providing a hierarchy context:

```
ElementInHierarchy('Structure as of 2015','Germany')
```

This way all elements of all hierarchies can be addressed.

The normal rules functionality is available, the following constructs are possible:

1. `[Products:OldHier:Total]` = something

   `[Products:NewHier:Total]` = something

   Multiple rules referring to different elements with the same name in the same dimension but different hierarchies.

2. `[Products:{OldHier:Total ,NewHier:Total}]` = something

   The two rules from a) are combined in one rule. The two elements have been put into a list. This functionality is already available.

3. `[Products:NewHier:Total] = [Products:OldHier:Total]`

   An element in one hierarchy can refer to an element in another hierarchy.

4. `[Products:OldHier:Total]` = something

   `[Products:OldHier:Total]` = something else

   If multiple rules refer to the same area, only the first rule is executed. In this case the target areas are the same, therefore the second one is never executed.

Hierarchies in MDX and schema requests

OLAP Server returns appropriate information in schemas that contain a HIERARCHY_NAME or HIERARCHY_UNIQUE_NAME column.

Each dimension in MDX and XMLA schema request has at least one hierarchy.

Currently OLAP Server allows only a single hierarchy of a dimension to be used for the calculation of a cell in the MDX statement. It is not possible to calculate a cell that refers to multiple elements of a dimension. In case the client sends an MDX statement that violates this rule, an error is returned.

The hierarchy unique names are the combination of the dimension name and the hierarchy name ([Dimension].[Hierarchy]). The only exception is the hierarchy with the same name as the dimension. This is the case for dimensions that get converted from older versions. For this hierarchy unique name is shorter as it contains only the dimension name ([Dimension]).

The unique names of elements generated by the OLAP Server contain the hierarchy unique name and the member part. OLAP Server resolves received unique names in the hierarchy noted in the unique name.

Hierarchies in the native modeling API

It is possible to send XML requests through a native connection in order to create hierarchies.

Representation of elements in the native API

The APIs of OLAP Server can be divided into two different groups:

- Interfaces that handle unique names: MDX and XMLA schema requests. Both interfaces are already prepared to handle hierarchies.
- APIs like IPO, XML Request or import files. Here two different ways to address elements are available.
  - Element IDs are used, for example by the dimension browser of the Excel Integration. Those IDs are unique inside the dimension, thus they identify an element and its hierarchy completely.
  - Elements are identified by their element names only. These element names might not be unique anymore, as another hierarchy can contain an element with a different content but the same name. Therefore elements have to be preceded with the hierarchy name. The separator between the hierarchy name and the element name is the <TAB> character. In some XML requests an optional hierarchy attribute is used to deliver the information. In the IPO interface SetHierarchy and GetHierarchy are added to handle the hierarchy information explicitly.

    If an element is not accompanied by hierarchy information the OLAP Server resolves it in the default hierarchy to keep the compatibility with existing clients.

Hierarchies and subsets

In version 11.0 subsets are restricted to contain elements from a single hierarchy only.

Hierarchies in LoadFromSource

It is possible to define both kinds of hierarchies - based on parent-child relationships and based on attributes - using Load From Source. Information about the hierarchies is stored in the new _Hierarchies table. Additionally the table _Parameters contains the information for the creation of attribute-driven hierarchies.

See "Hierarchies" on page 45 and "_Parameters" on page 210.

Hierarchies in Export AleaLog to the relational database

Using the value 5 for the ini-key `LogVersion` Export AleaLog to Relational Database supports 30 additional columns for the names of the hierarchies in the tables. If no hierarchy information is available, the content is NULL.

In case the `LogVersion` value is < 5, OLAP Server concatenates the hierarchy name and the element name and tries to put them to the element name column.

At server start OLAP Server checks if the `Alealog.txt` is compliant to the configured log version.

Hierarchies in the SQL interface

SQL Interface currently supports only the default hierarchy of a dimension.

Changed file structure

Due to the changes in cubes and dimensions the structure of the stored files has been changed:

- Older data models are converted into the new structure.
- Data models saved with the new structure cannot be loaded by older versions of OLAP Server.

Limits

Parameter	Max. number	Comment
Number of hierarchies per dimension	50	
Number of characters in hierarchy name/description	50/150	
Allowed characters in hierarchy name	Same as in dimension name	The name of a hierarchy must not start with a number. Exception: For backward compatibility implicit hierarchies (the hierarchy name is the same as the dimension name) may start with a number.
Number of elements in a dimension	10.000.000	This number includes all leaf members and the aggregated members of different hierarchies.
Number of parents for one child inside one hierarchy	84	A base element can have more than 84 members if it is used in multiple hierarchies.
Number of levels in a hierarchy	127	

Compatibility

- In the MDX/XMLA schema interface the default hierarchy was already present. No changes for existing data models. Clients which assume that OLAP dimensions have only one hierarchy have to adapt.
- In the native APIs the existing functions only consider the default hierarchy. Therefore it is recommended that existing clients continue working on converted data models.
- Functions have been extended or new functions have been created to support access to all hierarchies.

Hierarchies in XML modeling requests

Hierarchies are modeled through XML requests.

Parent-Child hierarchies

The XML based modeling requests support hierarchies on a dimension. Hierarchies can be defined as a part of the dimension:

```
<Alea:Request RequestID="1" Class="Dimension" Method="Write">
   <Alea:Dimension Name="Org" FirstBatch="true" LastBatch="true"
   LinesCount="1">
      <Alea:Description>Organization structure</Alea:Description>
      <Alea:Hierarchy Name="Org as of 2014">
         <Alea:Elements>
         N Fire Department
         N Police Department
         N Waste Removal
         C Emergency
         ! Emergency Fire Department
         ! Emergency Police Department
         C Standard
         ! Standard Waste Removal
         C All
         ! All Emergency
         All Standard
         </Alea:Elements>
      </Alea:Hierarchy>
      <Alea:Hierarchy Name="Org as of 2015 - create from scratch">
         <Alea:Elements>
         N Fire Department
         N Police Department
         N Waste Removal
         C All
         Fire Department
         Police Department
         Waste Removal
         </Alea:Elements>
         <Alea:Properties>
            <Alea:DefaultMember Name="All" />
            <Alea:LevelNames>
               <Alea:Level Number="1" Name="All" />
               <Alea:Level Number="0" Name="Departments" />
            </Alea:LevelNames>
         </Alea:Properties>
      </Alea:Hierarchy>
         <Alea:Properties>
            <Alea:DefaultHierarchy Name="Org as of 2015
            - create from scratch" />
         </Alea:Properties>
   </Alea:Dimension>
</Alea:Request>
```

The example creates a dimension Org with three hierarchies (Alea:Hierarchy). As no other dimension is referenced (Alea:DimensionTemplate) the dimension is built from scratch. One of the hierarchies is

defined to be the default one (Alea:DefaultHierarchy). Some base elements are used in both hierarchies, but the aggregated members differ.

The first hierarchy shows the organization structure as of 2014 before restructuring. It is defined by parent-child relationships. No other information is stored with the hierarchy than the name (Alea:Hierarchy/Name). The elements are defined in an import format (Alea:Elements).

The second hierarchy (Org as of 2015 - create from scratch) is created from scratch. The base elements are simply put under the All element. For the hierarchy some properties get defined (Alea:Properties). At first there is the name of the default member of the hierarchy (Alea:DefaultMember/Name). The names of the levels get defined as well (Alea:LevelNames).

Using DimensionTemplate an existing dimension can be changed or copied. In this case existing hierarchies can be modified or copied by using HierarchyTemplate. Or a hierarchy can be removed using the DeleteHierarchy tag.

```
<Alea:Request RequestID="1" Class="Dimension" Method="Write">
   <Alea:Dimension Name="Org" FirstBatch="true" LastBatch="true"
   LinesCount="1">
      <Alea:DimensionTemplate Name="Org">
         <Alea:DeleteHierarchy Name="Org as of 2015
         - create from scratch" />
      </Alea:DimensionTemplate>
      <Alea:Description>Organization structure</Alea:Description>
      <Alea:Hierarchy Name="Org as of 2015 - copy existing one">
         <Alea:HierarchyTemplate Name="Org as of 2014">
            <Alea:DeleteElement Name="Emergency" />
            <Alea:RemoveRelation Name="Waste Removal"
            Parent="Standard" />
            <Alea:DeleteElement Name="Standard" />
         </Alea:HierarchyTemplate>
         <Alea:Elements>
         ! All Fire Department 1
         ! All Police Department 1
         ! All Waste Removal 1
         </Alea:Elements>
         <Alea:Properties>
            <Alea:DefaultMember Name="Police Department" />
         </Alea:Properties>
         </Alea:Hierarchy>
         <Alea:Properties>
            <Alea:DefaultHierarchy Name="Org as of 2015
            - create from scratch" />
         </Alea:Properties>
   </Alea:Dimension>
</Alea:Request>
```

In this case the dimension Org is used as a template (Alea:DimensionTemplate). This contains the hierarchies that are to be deleted. In this case the hierarchy "Org as of 2015 - create from scratch" is deleted.

Use `IgnoreSourceNotExist=TRUE` within the `<Alea:DimensionTemplate>` element to specify that, if the source dimension does not exist, the template should be ignored and the dimension created from scratch.

For example:`<Alea:DimensionTemplate Name="Org" IgnoreSourceNotExist="TRUE">` .

Then a new hierarchy is created based on the first hierarchy (Alea:HierarchyTemplate/Name). Here two elements get deleted (Alea:DeleteElement/Name). Additionally the relations from Waste Removal to its parent is removed (Alea:RemoveRelation). This is not necessary as the element is removed anyway, but was added to show the functionality.

There is an explicit default member defined in the hierarchy (Alea:DefaultMember/Name).

The processing of the xml structure delivered with the `Dimension.Write` request works like this:

- Alea:DimensionTemplate is the first tag inside Alea:Dimension.
- The tags `Alea:DimensionTemplate` and `Alea:HierarchyTemplate` are only valid in the first batch.
- A hierarchy can be addressed multiple times in multiple batches.
- The dimension is committed after the last batch only.

Attribute-driven hierarchies

Creating a single attribute-driven hierarchy.

```
<Alea:Request RequestID="1" Class="Dimension"
Method="Write">
   <Alea:Dimension Name="Org" FirstBatch="true"
   LastBatch="true" LinesCount="1">
      <Alea:DimensionTemplate Name="Org" />
      <Alea:Hierarchy Name="By location"
      SourceHierarchy="Org as of 2014">
         <Alea:Description>Description of the hierarchy
         </Alea:Description>
         <Alea:TopLevel Element=" All departments" />
         <Alea:AttributeLevel TableID="0" AttributeName="CITY"
         EmptyValueElement="unknown location" />
         <Alea:BaseLevel Hidden="false" />
      </Alea:Hierarchy>
   </Alea:Dimension>
</Alea:Request>
```

The hierarchy is defined in multiple steps. At first the root level is defined (Alea:TopLevel). It contains the name of the top element (Alea:TopLevel/Element). This level is optional. A hierarchy without this level might have multiple root members.

This is followed by zero or more levels that are based on attributes (Alea:AttributeLevel). The necessary information is the number of the attribute table (Alea:AttributeLevel/TableID) and the name of the attribute (Alea:AttributeLevel/AttributeName). Optionally a name can be defined that is used in the case

that the attribute is not correctly defined (Alea:AttributeLevel/EmptyValueElement). This element name is used if the attribute value cannot be used as an element name. This can happen in multiple situations like invalid characters, relations that violate the tree structure or non-unique element names. If no attribute level is defined, the base elements are children of the TopLevel element - if that is defined. If the `EmptyValueElement` is not specified, or it is not usable, OLAP Server generates a name.

The members on the base level of this hierarchy (Alea:BaseLevel) are defined by one of the hierarchies created from parent-child relations. The name of that hierarchy is stored in this tag (Alea:Hierarchy/SourceHierarchy). In later versions it is planned to allow hiding the base elements (Alea:BaseLevel/Hidden). In version 11.0 this will not be possible. The only possible value is false.

Dimension.UpdateHierarchies

Attribute driven hierarchies are not updated automatically with every single attribute value change. This method should be called after all attribute values have been written. If it is not called, the hierarchy are not changed.

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="7" Class="Dimension"
   Method="UpdateHierarchies">
      <Alea:Dimension Name="DimName" />
   </Alea:Request>
</Alea:Document>
```

Example requests with responses

Creating a dimension with multiple hierarchies

Request

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1" Class="Dimension" Method="Write">
      <Alea:Dimension Name="Org" FirstBatch="true" LastBatch="true"
      LinesCount="1">
         <Alea:Description>D212</Alea:Description>
         <Alea:Hierarchy Name="Org as of 2014">
            <Alea:Elements
            N Fire Department
            N Police Department
            N Waste Removal
            N Driver License
            C Emergency
            ! Emergency Fire Department 1
```

```
            ! Emergency Police Department 1
            C Standard
            ! Standard Waste Removal 1
            ! Standard Driver License 1
            C All
            ! All Emergency 1
            ! All Standard 1
            </Alea:Elements>
         </Alea:Hierarchy>
         <Alea:Hierarchy Name="Org as of 2015">
            <Alea:Elements>
            N Fire Department
            N Police Department
            N Waste Removal
            N Driver License 2015
            C All
            Fire Department 1
            Police Department 1
            Waste Removal 1
            Driver License 2015 1
            </Alea:Elements>
         </Alea:Hierarchy>
         <Alea:Properties>
            <Alea:DefaultHierarchy Name="Org as of 2015"/>
            <Alea:DefaultMember Name="Police Department"
            Hierarchy="Org as of 2015"/>
         </Alea:Properties>
      </Alea:Dimension>
   </Alea:Request>
</Alea:Document>
```

Answer

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1">
      <Alea:Return>
         <Alea:Dimension Name="Org">
            <Alea:Hierarchy Name="Org as of 2014"/>
            <Alea:Hierarchy Name="Org as of 2015"/>
         </Alea:Dimension>
      </Alea:Return>
   </Alea:Request>
</Alea:Document>
```

Creating a hierarchy with template

Request

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1" Class="Dimension" Method="Write">
      <Alea:Dimension Name="Org" FirstBatch="true" LastBatch="true"
      LinesCount="1">
         <Alea:DimensionTemplate Name="Org">
            <Alea:DeleteHierarchy Name="Org as of 2014" />
        </Alea:DimensionTemplate>
         <Alea:Hierarchy Name="Org as of 2015 - copy existing one">
            <Alea:HierarchyTemplate Name="Org as of 2014">
               <Alea:DeleteElement Name="Emergency" />
               <Alea:RemoveRelation Name="Waste Removal"
               Parent="Standard" />
               <Alea:DeleteElement Name="Standard" />
            </Alea:HierarchyTemplate>
            <Alea:Elements>
            ! All Fire Department 1
            ! All Police Department 1
            ! All Waste Removal 1
            </Alea:Elements>
         </Alea:Hierarchy>
      </Alea:Dimension>
   </Alea:Request>
</Alea:Document>
```

Use `IgnoreSourceNotExist=TRUE` within the `<Alea:DimensionTemplate>` element to specify that, if the source dimension does not exist, the template should be ignored and the dimension created from scratch.

Response

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1">
      <Alea:Return>
         <Alea:Dimension Name="Org">
            <Alea:DimensionTemplate Name="Org">
               <Alea:DeleteHierarchy Name="Org as of 2014"/>
            </Alea:DimensionTemplate>
            <Alea:Hierarchy Name="Org as of 2016">
            <Alea:HierarchyTemplate Name="Org as of 2014"/>
            </Alea:Hierarchy>
         </Alea:Dimension>
      </Alea:Return>
   </Alea:Request>
</Alea:Document>
```

Error, in case of error in any element line

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1">
      <Alea:Return>
         <Alea:Dimension Name="Org">
            <Alea:DimensionTemplate Name="Org">
              <Alea:DeleteHierarchy Name="Org as of 2014"/>
            </Alea:DimensionTemplate>
            <Alea:Hierarchy Name="Org as of 2016">
               <Alea:HierarchyTemplate Name="Org as of 2014"/>
                  <Alea:Elements LinesCount="1">61
                  NewElement</Alea:Elements>
            </Alea:Hierarchy>
         </Alea:Dimension>
      </Alea:Return>
   </Alea:Request>
</Alea:Document>
```

Error in case of modified element does not exist

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1">
      <Alea:Return>
         <Alea:Dimension Name="Org">
            <Alea:DimensionTemplate Name="Org">
               <Alea:DeleteHierarchy Name="Org as of 2014"/>
            </Alea:DimensionTemplate>
            <Alea:Hierarchy Name="Org as of 2016">
               <Alea:HierarchyTemplate Name="Org as of 2014">
                  <Alea:DeleteElement Name="xyz">
                     <Alea:Error ErrorID="65"/>
                  </Alea:DeleteElement>
               </Alea:HierarchyTemplate>
            </Alea:Hierarchy>
         </Alea:Dimension>
      </Alea:Return>
   </Alea:Request>
</Alea:Document>
```

Error, in case delete hierarchy is not present

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1">
      <Alea:Return>
         <Alea:Dimension Name="Org">
            <Alea:DimensionTemplate Name="Org">
               <Alea:DeleteHierarchy Name="Org as of 2014">
                  <Alea:Error ErrorID="319"/>
               </Alea:DeleteHierarchy>
```

```
            </Alea:DimensionTemplate>
         </Alea:Dimension>
      </Alea:Return>
      <Alea:Error ErrorID="319" />
   </Alea:Request>
</Alea:Document>
```

Creating an attribute-driven hierarchy

Request

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1" Class="Dimension"
   Method="Write">
      <Alea:Dimension Name="Org" FirstBatch="true" LastBatch="true"
      LinesCount="1">
         <Alea:DimensionTemplate Name="PRODUCTS" />
         <Alea:Hierarchy Name="Org as of 2021"
         SourceHierarchy="PRODUCTS">
            <Alea:Description>Organization structure 2016
            </Alea:Description>
            <Alea:TopLevel Element=" All departments" />
            <Alea:AttributeLevel TableID="0"
            AttributeName="PRODMAN"
            EmptyValueElement="unknown location" />
            <Alea:Properties>
               <Alea:DefaultMember Name="Police Department" />
            </Alea:Properties>
         </Alea:Hierarchy>
      </Alea:Dimension>
   </Alea:Request>
</Alea:Document>
```

Use `IgnoreSourceNotExist=TRUE` within the `<Alea:DimensionTemplate>` element to specify that, if the source dimension does not exist, the template should be ignored and the dimension created from scratch.

Answer

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="1">
      <Alea:Return>
         <Alea:Dimension Name="Org">
            <Alea:DimensionTemplate Name="PRODUCTS"/>
         </Alea:Dimension>
      </Alea:Return>
   </Alea:Request>
</Alea:Document>
```

General hierarchies related XML requests

The examples above use the existing method `Dimension.Write`. The following methods have been implemented to maintain hierarchies.

Dimension.ListHierarchies

Returns the list of hierarchies defined for the given dimension

Request

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="7" Class="Dimension"
   Method="ListHierarchies">
      <Alea:Dimension Name="DimName" />
      </Alea:Request>
</Alea:Document>
```

Answer

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="7">
      <Alea:Return>
         <Alea:Dimension Name="Org">
            <Alea:Hierarchy Name="Org" HierarchyType="Parent/Child">
               <Alea:Description>Hierarchy1</Alea:Description>
         </Alea:Hierarchy>
         <Alea:Hierarchy Name="By location"
         HierarchyType="Attribute-driven">
            <Alea:Description>Hierarchy2</Alea:Description>
         </Alea:Hierarchy>
         <Alea:Hierarchy Name="newADH"
        HierarchyType="Attribute-driven">
            <Alea:Description>Hierarchy3</Alea:Description>
         </Alea:Hierarchy>
         <Alea:Hierarchy Name="Org as of 2016"
         HierarchyType="Parent/Child">
            <Alea:Description>Hierarchy4</Alea:Description>
         </Alea:Hierarchy>
         </Alea:Dimension>
      </Alea:Return>
   </Alea:Request>
</Alea:Document>
```

Dimension.GetHierarchyDefinition

Returns the hierarchy definition of a specified OLAP dimension.

Request

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="7" Class="Dimension"
   Method="GetHierarchyDefinition">
      <Alea:Dimension Name="DimName" >
         <Alea:Hierarchy Name="NameOfHierarchy" />
      </Alea:Dimension/>
   </Alea:Request>
</Alea:Document>
```

Deleting a hierarchy

Hierarchies are deleted using a `Dimension.Write` request. The `Alea:DeleteHierarchy` tag is sent inside the `Alea:DimensionTemplate` tag.

Request

```
<Alea:Request RequestID="1" Class="Dimension" Method="Write">
   <Alea:Dimension Name="Org" FirstBatch="true" LastBatch="true"
   LinesCount="1">
      <Alea:DimensionTemplate Name="Org">
         <Alea:DeleteHierarchy Name="Org as of 2015
         - create from scratch" />
      </Alea:DimensionTemplate>
     </Alea:Dimension>
</Alea:Request>
```

Dimension.PutProperties

Request

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="001" Class="Dimension"
   Method="PutProperties">
      <Alea:Dimension Name="DimensionName">
         <Alea:Properties>
            <Alea:DefaultMember Name=""/>
            <Alea:LevelNames>
               <Alea:Level Number="0" Name="Cities"/>
               <Alea:Level Number="1" Name="States"/>
            </Alea:LevelNames>
            <Alea:DefaultHierarchy Name="Hierarchy A"/>
            <Alea:Hierarchy Reverse="true"/>
            <Alea:FlatView Enabled="true"/>
            <Alea:ODBOType Code=""/>
            <Alea:MemberKey AttribTableID="" FieldName=""/>
```

```
            <Alea:MemberCaption AttribTableID="" FieldName=""/>
            <Alea:AccessCube Name=""/>
            <Alea:Translation LocaleIdentifier="7">
               <Alea:Description>Description Dimension
               </Alea:Description>
               <Alea:Caption>Caption Dimension</Alea:Caption>
               <Alea:MemberCaption>MemberCaption</Alea:MemberCaption>
               <Alea:MemberCaption AttribTableID="1"
               FieldName="CaptionDE"/>
               <Alea:LevelCaption>LevelCaption</Alea:LevelCaption>
            </Alea:Translation>
            <Alea:Hierarchy Name="Hierarchy A">
               <Alea:Properties>
                  <Alea:DefaultMember Name="Total"/>
                  <Alea:LevelNames>
                     <Alea:Level Number="0" Name="Cities"/>
                     <Alea:Level Number="1" Name="States"/>
                  </Alea:LevelNames>
                  <Alea:Translation LocaleIdentifier="7">
                  <Alea:Caption>Caption Hierarchy</Alea:Caption>
                  <Alea:Description>Description Hierarchy
                  </Alea:Description>
                     <Alea:LevelCaption Level="0">Jahr
                     </Alea:LevelCaption>
                     <Alea:LevelCaption Level="1">Quartale
                     </Alea:LevelCaption>
                     <Alea:LevelCaption Level="2">Monate
                     </Alea:LevelCaption>
                  </Alea:Translation>
               </Alea:Properties>
            </Alea:Hierarchy>
         </Alea:Properties>
      </Alea:Dimension>
   </Alea:Request>
</Alea:Document>
```

Dimension.GetProperties

```
<Alea:Document xmlns:Alea="http://www.misag.com">
   <Alea:Request RequestID="001">
      <Alea:Return>
         <Alea:Properties xmlns:Alea="http://www.misag.com">
             <Alea:DefaultHierarchy Name="Org as of 2015"/>
             <Alea:Translation LocaleIdentifier="0">
                <Alea:Description>D212</Alea:Description>
             </Alea:Translation>
             <Alea:ODBOType Code="3"/>
             <Alea:DefaultMember Name="All"/>
             <Alea:Hierarchy Name="Org as of 2014">
```

```
                <Alea:Properties xmlns:Alea="http://www.misag.com">
                   <Alea:DefaultMember Name="All"/>
                   <Alea:LevelNames>
                     <Alea:Level Number="1" Name="AllHir1"/>
                     <Alea:Level Number="0" Name="Department"/>
                  </Alea:LevelNames>
                  <Alea:Translation LocaleIdentifier="7">
                     <Alea:Description>hier dies</Alea:Description>
                     <Alea:LevelCaption
Level="0">Jahr</Alea:LevelCaption>
                     <Alea:LevelCaption
Level="1">year</Alea:LevelCaption>
                 </Alea:Translation>
                 <Alea:Translation LocaleIdentifier="0">
                    <Alea:Description>Hierarchy1</Alea:Description>
                 </Alea:Translation>
              </Alea:Properties>
           </Alea:Hierarchy>
            <Alea:Hierarchy Name="Org as of 2015">
               <Alea:Properties xmlns:Alea="http://www.misag.com">
                  <Alea:DefaultMember Name="All"/>
                  <Alea:LevelNames>
                     <Alea:Level Number="1" Name="AllHir2"/>
                     <Alea:Level Number="0" Name="BaseElement"/>
                  </Alea:LevelNames>
                  <Alea:Translation LocaleIdentifier="0">
                     <Alea:Description>Hierarchy2</Alea:Description>
                  </Alea:Translation>
               </Alea:Properties>
            </Alea:Hierarchy>
         </Alea:Properties>
      </Alea:Return>
   </Alea:Request>
</Alea:Document>
```

Changes/additions to native functions

The functions described in this section have been changed to support hierarchies.

Cube.ImportCells

The previous syntax addresses the default hierarchy for all dimensions. If the client adds the attribute `'WithHierarchies="true"'`, the server expects a hierarchy and an element column instead of the element column alone for each dimension. Empty hierarchy columns address the default hierarchy.

Cube.GetCellNote, Cube.PuCellNote, Cube.DeleteCellNote

The `Element`-tag can take an additional attribute 'Hierarchy' to address a non-default hierarchy.

Cube.Read

In the single-value variant with a `CellCoordinates` tag having an `Elements` attribute can take an additional `Hierarchies` attribute listing the target hierarchies separated similar to the elements.

The data-query based variant receives additional hierarchy columns (as the expected input in `Cube.ImportCells`) in two cases:

- The underlying dataarea explicitly addresses a non-default hierarchy.
- The `DataQuery` tag has the attribute `'WithHierarchies="true"'`.

In both cases the returned `CellBatch` tag is marked through `WithHierarchies="true"`.

Cube.DataareaCopy

The element specifications can have a `Hierarchy` attribute to address a non-default hierarchy. For the source of an element-to-element mapping the `SourceHierarchy` attribute is used.

Cube.DataareaDefine

Selectors and elements can carry a `Hierarchy` attribute to target a non-default hierarchy with the definition.

Cube.DataareaCalculateHash

If the underlying dataarea addresses non-default hierarchies the hash is calculated using the concatenated hierarchy and element name instead of using only the element name, if they are part of the hash.

Hierarchies in OLAP Administration

In the OLAP Administration you can define parent-child hierarchies and attribute hierarchies.

1. Right-click a dimension and select **Hierarchies**.
2. Using the Hierarchies dialog, you can:
   - Create a parent-child hierarchy
   - Create an attribute hierarchy

- Update hierarchies (after you have edited the hierarchy)

Parent-Child Hierarchy editor

To create a parent-child hierarchy:

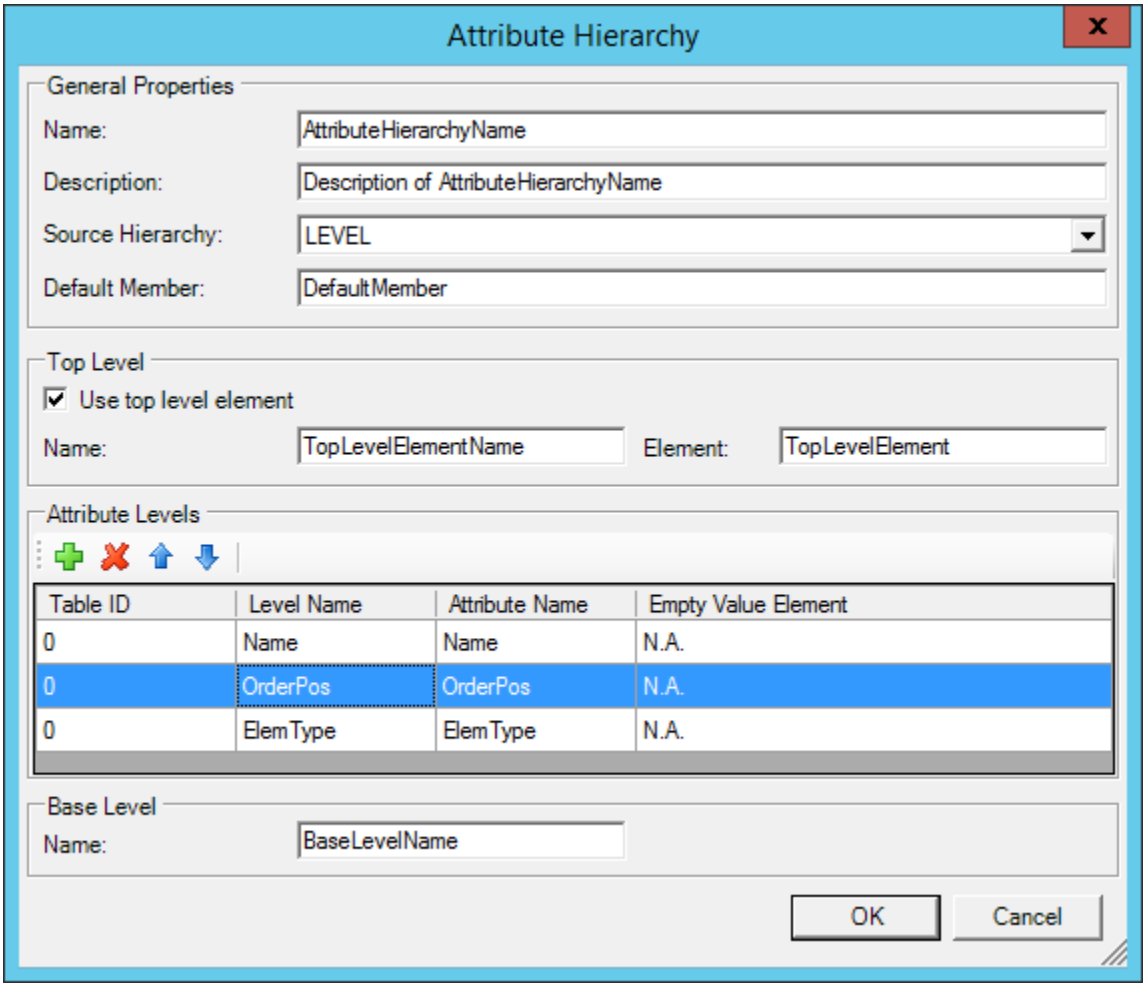
1. Select **Create Parent-child Hierarchy** in the Hierarchies dialog. The Parent-child Hierarchy editor opens.
2. Enter a name and a description.
3. Click the **Load** button to open the Select Dimension dialog.
4. Select a dimension and click **OK**.
5. Edit the dimension and click **OK**. These are the edit options for the Parent-child hierarchy editor:

Button		Description
	Load	Displays the Select Dimension dialog where you can select one dimension from the available dimensions.
	Save	Saves the hierarchy.
	Properties	Displays the Properties dialog where you can select the Display options.
	Show Top-Level Elements	Display all elements which have no parents (usually the top-level elements of the dimension).
	Show All Elements	Display all the elements of the dimension in hierarchical or non-hierarchical order, depending on the display order.
	Hide Lowest Level	Hide all elements at the lowest level currently displayed.
	Show Next Level	Show all children of the elements currently displayed.
	Base Element	Changes the element type of the selected element to base element.
	Dimension Rule	Changes the element type to a dimension rule.
	String	Changes the element type to string.

Button		Description
	New Child	Creates a child element.
	New Sibling	Creates a sibling element.
	Edit	
	Delete	Opens the Delete Elements dialog. You can choose if only the selected or the selected elements including their children are deleted.
	Remove	Removes the selected element from the parent element.
	Find	Search for elements or attributes and marks those elements that match the search.
	Next, Previous	Enabled after a search. Use the Next and Previous buttons to scroll through the result of the search.
	Cut, Copy, Paste	Cuts/Copies and pastes elements including their children.
	Indent Left, Indent Right	Increases/Decreases the indent.
	Move Up, Move Down	Moves up/down the element one level.

Attribute Hierarchy editor

To create an attribute hierarchy, select **Create Attribute Hierarchy** in the Hierarchies dialog. The Attribute Hierarchy editor opens.



Edit options of the parent-child hierarchy editor:

General Properties	Name	Specify the name of the hierarchy.
	Description	Specify the description of the hierarchy.
	Source Hierarchy	Select the source hierarchy from the Source Hierarchy drop-down list.
	Default Member	Specify the default member of the hierarchy. The default member is the element which is displayed in list in a report by default.

Top Level (optional)	Name, Element	Specify the top level name and the element name. **Note:** The top level element is added to the existing levels, the number of levels is increased by one.
Attribute Levels	: Add Attribute	Adds an attribute. Opens the Attributes dialog where you can select one or more attributes.
	: Remove Attribute	Remove the selected attribute.
	, : Move Up, Move Down	Reorder levels.
Base Level	Name	Specify the name of the base level.

Number formats for measures

5

One dimension of every OLAP cube is declared as a measure dimension. For example, in the Best Practices sample database, the Analysis cube contains a dimension called Measure. Each element of the Measure dimension is a measure, such as Gross Margin, Revenue, Units, and Discounts.

You can apply an appropriate number format to each measure. For example, format discounts as percentages or apply a currency format to unit prices. In any Application Studio or Office Plus report in which a measure is used, the value of the measure is automatically displayed in the appropriate format.

The formats for each measure must be stored in an attribute called **Format_String** on the measure dimension. OLAP Server passes the value as a cell format with the MDX result set.

The formats can be standard spreadsheet formats such as `0.00%` or `#,##0.00` or they can be the names of styles. For example, if you have previously created a style called Percent in Application Studio you can specify **Percent** as a value of the Format_String attribute.

In Application Studio reports, the cells in which measures are displayed must be formatted with the tilde `(~)` format.

See *Specifying number formats* in theApplication Studio online help.

You can use format strings in calculations. For example, if you have a calculation in Application Studio called Discounts in % you could specify **0.00%** as the `Format String` property in the Advanced pane of the server list designer.

Regional settings on the client computers have an impact on number formats. For example, if you use standard currency formats, the dollar sign is a euro sign on client computers in Europe.

Creating the format_string attribute

To apply different formats for each measure in the measure dimension, the dimension must have an attribute called **Format_String**. The attribute name is case-insensitive but must include the underscore character between Format and String.

1. In OLAP Administration, expand the Favorite Database that contains the required measure dimension.
2. Right-click the measure dimension and select **Attribute Tables**.

3 Select a tab which does not relate to an existing attribute table and click **Create**.
4 Specify this information:

Name
Specify **Format_String**.

Description
In the Dimension Browser, the description is displayed as the column name for the attribute. So, it can be helpful to use Format_String as the description too.

Type
Select **String**.

Width
Specify the maximum number of characters that your numerical formats can contain.

5 Press Enter and then click **Cancel**.
6 Click **Apply** and then click **Yes** to confirm that you want to create the attribute table.
The attribute is created. Attribute tables are empty when you create them.

Specifying formats in the format_string attribute

The Format_String attribute is empty when you create it.

To specify formats for each element of the measure dimension:

1 In OLAP Administration, expand the Favorite Database that contains the required measure dimension.
2 Double-click the measure dimension.
3 In the Dimension Browser, select the attribute table that contains the Format_String attribute.
4 Select the measure to format.
5 Click **Edit Attribute Record**.
6 Specify the required number format.
7 Click **OK**.

Setting up the OLAP Server

6

Components

There are two components to every OLAP Server: the Communication Manager and the OLAP Server. In OLAP Server the executable files of the server are started by the OLAP Server Communications Manager. The OLAP Server is not visible on the server computer by default, only the Communication Manager. The administration of the server is done either through the OLAP Administration, the initialization files, or the client. For additional information on this component, refer to its documentation.

Communication layer

One major goal of OLAP Server is to load multiple databases on one computer. The equipment of a server is mostly based on whether the computer is fast enough to answer requests in reasonable time during peak periods. But this also means that the server is oversized for periods outside these peaks. Through the possibility of running multiple databases on one computer, the administrator can take advantage of the free resources in non-peak-periods, but still keep different data models separated from each other. This means that a user who has rights for one database still has to log on to a second one, if he wants to see its data. The same applies to the administrators: they can change and see everything on their databases, but need an account, if they want to access another database on the same server.

Purpose of the Communication Manager

To run databases separately, each database is handled by one OLAP Server in a separate process. Therefore, it is no longer sufficient for a client to know the name of the server computer where the process is running, because there can be several OLAP Server processes running on the same server computer. Now, when a user wants to use the database 'Tutor' on computer 'MISSRV' the client must connect to 'MISSRV/Tutor'. To provide this and other internal information, such as the TCP/IP port, OLAP Server Communication Manager is used.

OLAP Server Communication Manager contains information about the databases that are available and may be accessed by a client. When running the databases locally and accessing them via a

standalone client, an available database does not need to be started. The standalone version of OLAP Server loads databases on demand. When an available database is logged on to from the OLAP Server dialog, OLAP Server Communication Manager starts the requested server locally and loads the database. In the case of the remote server, this is not true. When starting the OLAP Server Communications Manager on a remote server, all servers/databases that are to be made available to remote clients must be loaded.

OLAP Server Communication Manager determines the list of available databases from an initialization file stored on the computer on which the Communication Manager is located. This gives the administrator of that computer control over which data models are available. This initialization file also contains the instruction to start the database automatically when loading the Communication Manager (necessary when running a remote OLAP Server), or manually upon the first attempt to connect to the database (standalone version only).

On each computer there can only be one OLAP Server Communication Manger. In normal operation, this program runs as an NT Service. It is important to know that, once communication has been established, the clients are directly connected to the server. Client requests are not sent through the OLAP Server Communication Manager, but are sent directly to the server.

Communication between Infor BI OLAP Server Communication Managers

As stated above, the client retrieves a list of available databases from an OLAP Server Communication Manager. If the OLAP Server Communication Manager only knows about the databases available on its own computer, this is not very efficient. Therefore, a mechanism has been implemented allowing the Communication Managers to exchange information. There are some options to control this process. The most important one is that the administrator of the computer may define whether the OLAP Server Communication Manager shares information with all, none, or only some explicitly defined OLAP Server Communication Managers in the network.

The Communication Manager uses the internet protocol UDP/IP for communication.

Event processing

OLAP Server allows external programs to register themselves as receiver of events with an OLAP Server. Events are generated automatically by the server, whenever clients perform certain actions. For example, changing a cell value or a dimension. The server adds additional information, for example, about the user who preforms this action, to the event and sends the information to every event receiver who subscribed to that event. Once the external program received an event, it is up to the program to process and react to the event. An example of an external event-processor is the Event Agent, a component of Infor BI.

Besides the automatically generated events, there are events that can be raised by clients. The generic event is accompanied by a string containing any information to be transferred. Once the event is

accepted by an event-receiver, it is up to this external program to evaluate and react to this event. OLAP Server does not evaluate this string, it just passes it through. To use the new events they must be activated at the server.

See "Using the initialization files" on page 261 for information on how to set the key EnableEvents.

The event receiver must use the OLAP Server component EventServerConnectivity (`mdsESC.dll`) that provides the functionality to receive events.

See "Event Agent" on page 111).

Database structure

Server naming convention

Since it is possible to address several models on one server, you must refer to a model by a combination of server and model name. For example, if you have two models, 'TUTOR' and 'GENESIS' loaded on your LOCAL server, use:

- "LOCAL/TUTOR"

  as the 'Server' argument in an Excel formula or API function calling information from the TUTOR model, and
- "LOCAL/GENESIS"

  as the 'Server' argument in an Excel formula or API function calling information from the GENESIS model. The slash is used to separate the server name from the model (database directory) name.

When running several models/databases on one server, each model must be logged-on to separately. A separate instance of OLAP Server is started for each model.

Loadable databases for a local server

As noted before, all models for an OLAP Server (LOCAL or Application) must be stored in a specified root directory. There are three initialization files (`Alea.ini`, `Db.ini`, `Mdsex.ini`) in which parameters for the operation of the OLAP Server and Office Plus are set. The `Mdsex.ini` file controls the settings for Office Plus, it is located in the Windows directory of the client computer. The `Alea.ini` file controls the settings for the OLAP Server (local or remote). It is located in the `\ProgramData\Infor\BI\OLAP` directory of the computer on which OLAP Server is running. The `Db.ini` file contains settings for a particular database/application. It is located in the directory of the database/application. All parameters for these initialization files may be changed in the OLAP Administration (installed with OLAP Server) or from within Office Plus.

See "Using the initialization files" on page 261.

`Mdsex.ini` contains the Office Plus settings. All settings in this `ini` file are set in the OLAP Server Options dialog in Office Plus. The settings for each database (for example, whether SMP, or rules caching is used, which cubes are loaded automatically and so on.) are specified in the `Db.ini` file. Settings for communications, locations of databases, default databases etc., are specified in the `Alea.ini` file. The parameters for the `Db.ini` file and the `Alea.ini` file are set in the OLAP Administration.

Configuration of databases can be performed for each registered database individually or for all databases on the local computer and any remote computer. The components Database Settings, Scheduler Settings, Protocol Settings, and Event Agent Settings can be found in the container of every registered database in the favorites list (used for the individual configuration) or below the Local Computer component for global configuration. Underneath Local Computer you can also change the settings concerning the OLAP Server Communication Manager, the communication in general, logging, and administer the database.

The General component specifies parameters in the `Alea.ini` file. The other components specify parameters in the `Db.ini` files of individual databases or in the general `Db.ini` located in the database root (`DBRoot`) directory. For an individual database, OLAP Server looks first at the settings in the `Db.ini` of that database. If a parameter is not set there, OLAP Server looks for the settings in the general `Db.ini` located in the `DBRoot`.

Defining loadable databases

You can change the location of the LOCAL database root directory using the OLAP Administration. The default location of the database root directory is `DBRoot=CC:\Users\Public\Documents\Infor\BI\OLAP\Data` which is set during installation.

When the LOCAL database root path is stored on a network drive, it is important to note that each new database created is a subdirectory of the database root. Therefore, the user must have network access rights that allow creation of subdirectories on that drive.

Each database/model must have loading instructions. If a database/model is specified to load automatically, the model is loaded automatically when OLAP Server is started. Specified cubes in the database/model are loaded into the memory when the server is started. If the database/model name is specified to load manually, the cubes in the database are only loaded when a client references the database/model.

Note: Loadable databases for a remote server are defined in the same way as loadable databases for a LOCAL server.

Protocol used by OLAP Server

A protocol is the communication software that allows the client and the server to communicate via a Local Area or Wide Area Network (LAN/WAN). OLAP Server uses the TCP/IP communication protocol.

You must enable the TCP/IP protocol to use OLAP Server as a client–server software. This is done using the Communication component in the OLAP Administration.

See "Using Infor BI OLAP Administration" on page 73.

Note: Starting with Infor BI 10.5.0 OLAP Server supports IPv6.

Default ports

By default the OLAP Server and its components uses these ports:

Component	Port	Type
OLAP Server	2904	UDP, TCP
Communication Manager Listen Port Secure Port	8210 8211	UDP, TCP

Using Infor BI OLAP Administration

Starting Infor BI OLAP Administration

OLAP Administration is installed with OLAP Server. It enables the administrator to configure server and client without using the Excel Client interface. The entries are stored in the initialization files. Start OLAP Administration via **Start > All Programs > Infor Business Intelligence > OLAP Administration**.

Note: If you want to start OLAP Administration on a 64-bit operating system manually via the mmc command, use the 32-bit version of MMC (MMC32).

```
mmc "C:\Program Files (x86)\Common Files\Infor\BI\Infor BI OLAP
Administration.msc" /32
```

Configuration

Configuration of databases can be performed for each registered database individually or for all databases on the local computer and any remote computer. The components Database Settings, Scheduler Settings, Protocol Settings, and the Event Agent Settings can be found in the container of

every registered database in the favorites list (used for the individual configuration) or below the Local Computer component for global configuration. Underneath Local Computer you can also change the settings concerning the OLAP Server Communication Manager, the communication in general, logging, and administer the database.

General

The choices specified in this component are written to the `Alea.ini` and `Db.ini` files as parameter settings.

The language settings are written to the [INIT] section of the `.ini` file.

Option	Description	Values	Default value
Language OLAP Server and VBAPI	Select the language for OLAP Server and VBAPI.	001 - English, 007 - Russian, 031 - Dutch, 033 - French, 034 - Spanish, 039 - Italian 042 - Czech,046 - Portuguese (Brazil), 049 - German, 081 - Japanese, 086 Chinese (Simplified)	English (001)
Language OLAP Administration	Select the language of OLAP Administration.	001 - English, 007 - Russian, 031 - Dutch, 033 - French, 034 - Spanish, 039 - Italian 042 - Czech,046 - Portuguese (Brazil), 049 - German, 081 - Japanese, 086 Chinese (Simplified)	English (001)

Option	Description	Values	Default value	`ini` file key
Root directory of the program files	The root directory of the program files	File system path	Set by setup	[INIT] ProgramRoot
Database root directory	The root directory of the databases	File system path	Set by setup	[INIT] DBRoot

The modified settings only take effect after restarting the Administration Console.

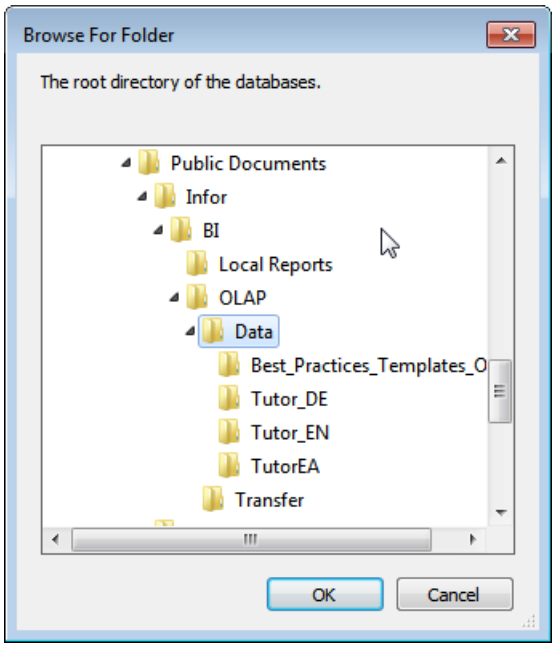
You can also set the database root for the OLAP Server by editing the `Alea.ini` file in the `WINDOWS` directory. The default location of the database root directory, which is set during installation, is `C:\Users\Public\Documents\Infor\BI\OLAP\Data`.

See "Using the initialization files" on page 261 and "Alea.ini" on page 265.

To change the path of the database root

1 Select **Database root** and click the ellipsis button.

2 The Browse for Folder dialog box opens. Choose the directory and click **OK**. All databases to be loaded by the server must exist as subdirectories of the `DB Root` directory.



Note: If you use a network directory, network access must include read/write access to this directory and the ability to create subdirectories.

In the above case, the root directory is `C:\Users\Public\Documents\Infor\BI\OLAP\Data\`. Three databases have been defined as available for access, all are specified to load cubes and dimensions manually. Database Settings

The Database Settings component allows you to define your preferences for the databases.

Note: When you use the OLAP Administration to change global database-settings, a `Db.ini` is created in the database root-directory. When you change specific database settings, a `Db.ini` is created in the database directory.

Most database settings such as rule cache size, dimension page size etc., are saved in the `Db.ini`. A `Db.ini` file can be present in all OLAP Server Databases/Models created or accessed by OLAP Server. A `Db.ini` file may also be located in the database root and used with all databases. If a parameter is not found in the `Db.ini` file in the database directory, the server will look for the `Db.ini` file in the database root to retrieve the setting. If a `Db.ini` file is not present in the root directory or in a database directory, the default settings for the `Db.ini` file are used.

See "Using the initialization files" on page 261.

The settings marked by an asterisk (*) can be applied to the database at runtime. Click the **Apply runtime settings** command on the shortcut menu of the database to execute the settings at runtime.

This table describes the general settings options:

Option	Description	Values	Default value	`ini` file key
Save all parameters in the `Db.ini` file	All parameters for the administration of databases are displayed in the `Db.ini` file	Yes, No	Yes	[INIT] FullContent
Default access rights to data*	Default access rights to dimension elements or cubes, if the TABACC (Table Access Control) and/or the DAC (Dimension Access Control) is enabled.	None, Read, Write, All	All	[INIT] DefaultHandling
Visible Server	Select YES to display the console window, even if the server was started by the Communication Manager.	Yes, No	No	[INIT] Server
Restrict caching (only for individual database setting)				This parameter is obsolete starting with OLAP Server 10.3.0.
Default startup mode for new databases*	Defines the start mode of new databases.	Automatic, Automatic + Events, Manual, Manual +Events, Administrator-Controlled Mode	Manual	Alea.ini [INIT] DefaultStartup-Mode
Startup mode	Defines the start mode of the current database.	Automatic, Automatic + Events, Manual, Manual +Events, Administrator-Controlled Mode	Manual	Alea.ini [INIT] DefaultStartup-Mode
Check cubes*	Specifies, when the cube integrity is checked. This check compares the cube in memory to the cube on disk. If inconsistencies are found, they are re-	L, S, M or any combination thereof O (Optimize): this value is obsolete starting with Infor PM OLAP 10.0.	-	[MEMORY] CheckTables

Option	Description	Values	Default value	`ini` file key
	paired automatically. • L: upon loading • S: upon saving • M: manual			
User who is allowed to connect to a database during the start of the Event Agent	Name of the user who may connect to the database during the reservation mode when the Event Agent is started.			[INIT] AllowedUserAtStartup
Case sensitivity for attribute values	YES: Attribute value tests are case sensitive (default value until 5.2.0) NO: Attribute value tests are not case sensitive Applies only to searching in dimension browser and attribute driven subsets.	YES, NO	NO	[INIT] AttributesCaseSensitive

Warning: We recommend you run OLAP Server as a service in a production environment and not as a Visible Server. The closing behavior in Microsoft Windows operating systems (starting with Windows Vista) kill the server process after ca. 5 seconds when the "Close Window" event is detected. Therefore it cannot be assured that all cubes are saved correctly.

This table describes the Rules Engine options:

Option	Description	Values	Default value	`ini` file key
Maximum calculation time	Maximum calculation time for a data call that is restricted by a pending write request.	1 - 3600	3600	[INIT] MaxCalcTime
Calculated values cache *	Cache for calculated values (rules calculated and consolidated values). '0' disables the cache, any value	0 - 100000	1	[MEMORY] CacheSize

Option	Description	Values	Default value	ini file key
	greater than 0 enables it.			

This table describes the communication options:

Option	Description	Values	Default value	ini file key
Listen Port TCP	TCP port by which the server can be addressed via network. If no value is specified, the server selects automatically a free port. The specification of a TCP port makes only sense, if you want to release just that port on a firewall.		2904	Alea.ini [TCPIP] ListenPort
Listen Port HTTP	Port on which the OLAP Server listens for HTTP XMLA Requests	integer	8210	[HTTP] ListenPort
Listen Port HTTPS	Port on which the OLAP Server listens for HTTPS XMLA Requests	integer	8211	[HTTP] SecurePort
HTTP/S session timeout*	Length of timeout for HTTP(S) based connections in seconds	60...5400	900	[HTTP] SessionTimeOut
Network Server	This database is accessible in the network.	Yes, No	No	[INIT] NetServer

This table describes the Infor BI Repository options:

Option	Description	Values	Default value	ini file key
Common Object Store Registration	Name of the registration to be used in Repository.		Best_Practices_Templates	[COS] Repository
Project name	Name of the project to be used in Repository. Note		Best Practices Templates	[COS] ProjectName

Option	Description	Values	Default value	`ini` file key
	that one OLAP Server database retrieves the permissions and roles from one OLAP Permission Management in the Repository, not vice versa.			
OLAP Server Permission Management	Alias for this database model in the OLAP Server Permission Management of the Repository.		<Database>	[COS] ModelName

This table describe the multi-processor settings options:

Option	Description	Values	Default value	`ini` file key
Number of processors	Set this parameter for a slightly improved performance. Even with default settings OLAP Server works fine on a multi-processor computer.	1 - ...	9999	[INIT] ProcessorCount
Processor affinity mask	With this mask you instruct the operating system to run the OLAP Server on certain processors only. For additional information refer to the documentation of the WIN API 32 command SetProcessorAffinityMask which can be found in the documentation of the operating system.	0x00000001 - 0xFFFFFFFF		[INIT] ProcessAffinity-Mask

This table describes the splashing options:

Option	Description	Values	Default value	`ini` file key
Undo files directory	Directory to store undo files created during splashing.		Database directory	[INIT] UndoFilesRoot
Maximum number of cells for one splashing operation	Maximum number of basic cells to be filled with one splashing operation	0 - 5000000	100000	[INIT] MaximumSplash-Values
Allow rules during splashing	NO : The algorithm will not allow rule calculations in cells changed by the algorithm or in the external source cell. YES : The algorithm will not check for cells calculated by rules. Due to this the calculated value may differ from the one entered by the user.	Yes, No	No	[INIT] AllowSplashOn-RuleCells

This table describes the memory options:

Option	Description	Values	Default value	`ini` file key
Size of the calculation block	Maximum number of cells used for internal rule calculation. Use scientific notation to avoid problems with decimal separator. For example, 1E6 instead of 1000000.	1 - 1e100	1E80	[INIT] MaxTileSize
Calculation slots	Number of slots used for calculation - each calculated slice needs one slot. Each rule cell reference needs one. This number limits the amount of requests calculated simultaneously and	256 - 16384	1024	[MEMORY] CSDSlots

Option	Description	Values	Default value	`ini` file key
	the recursion depth. 1 slot represents about 10kB of computer memory.			

This table describes the cube loading options. These options are only for individual databases.

Option	Description	Values	Default value	`ini` file key
Automatic cube loading	Specifies which cubes is loaded immediately after the start of the server.	All, None, Listed	None	[INIT] LoadTables
List of automatically loaded cubes	List of cubes to be loaded automatically.	Cube name		[LoadTables] Table1

This table describes the Provider options:

Option	Description	Values	Default value	`ini` file key
Format of unique names	Defines the format of unique member names. (Long: <dimensions name>.<all ancestors even non-visible>.<element name>. For example, [PRODUCT].[Total].[Total Monitors].[ProView VGA 12]; short: <dimension name>.<member name>.<number of occurrence>. For example, [PRODUCT].[ProView VGA 12].[1]).	Long, Short, Reduced	Long	[PROVIDER] UniqueNameFormat
Remove copies in OLAP subsets	YES: only the first occurrence of an OLAP Server member during the conversion of OLAP subsets to Named Sets.	Yes, No	No	[PROVIDER] RemoveDuplicatesInAleaSubsets

To make a database available in the network, the TCP/IP protocol must be enabled, and in the Database Settings component, the option Network server must be set to Yes.

Caution: If you want to use both OLAP Server and rules, set the rules cache-size in either the global database-settings or the specific database-settings. Otherwise, the rules cache is not used and performance is reduced.

Scheduler Settings

The Scheduler Settings component provides options for scheduled events to occur on the server. The table of the Administration Console on the right displays the options in the left column and lists or entry fields for the settings in the right column. In the lower part of the table you find brief descriptions of the options (which are displayed when you click them).

Backup copies the entire directory with all the files to another directory.

This table describes the general setting option:

Options	Description	Values	Default value	`ini` file key
Activate scheduler*	Activates the scheduler.	Yes, No	Yes	[INIT Scheduler]

This table describes the server saving options:

Options	Description	Values	Default value	`ini` file key
Scheduled server saving*	The server saves data periodically.	Yes, No	Yes	[Scheduler] SaveServer
First save*	First point in time for data storage.	00:01 - 23:59, NOW	15:00	[Scheduler] FirstSave
Save interval*	Interval for data storage.	00:01 - 23:59	01:00	[Scheduler] SaveInterval
Scheduled backup of the server*	Server performs periodical data backup.	Yes, No	No	[Scheduler] BackupServer
Delete content of the backup directory	Deletes the content of the backup directory. Warning: If set to YES, the content of the backup directory is deleted before each backup.	Yes, No	Yes	[Scheduler] EmptyBackupTargetDir

Options	Description	Values	Default value	`ini` file key
First backup*	First backup. Apart from time values, you can also enter Now: the backup will be carried out some minutes after the start of the server. The format is HH:MM.	00:01 - 23:59, NOW	15:00	[Scheduler] FirstBackup
Backup interval*	Backup interval The format is HH:MM.	00:01 - 23:59	01:00	[Scheduler] BackupInterval
Backup path*	Path to the directory for the backup file.		<database directory>\BACKUP	[Scheduler] BackupPath
Backup numbering	YES: the suffix 'nnnn' is attached to the backup path to keep multiple backup directories. 'nnnn' is the number stored in BackupNextNumber with leading zeros.	YES, NO	NO	[Scheduler] BackupNumbering
Next back up number	Next backup number. The server increases the backup number after every backup, no matter if the backup operation was scheduled or manually started.	<0,9999>	0	[Scheduler] BackupNextNumber
Maximum backup Number	If the backup number exceeds the specified maximum backup number, the Next backup number is set to 0. If -1 is set, the backup number is unlimited.	<-1,9999>	-1	[Scheduler] BackupMaxNumber

This table describes the inactive users options:

Options	Description	Values	Default value	`ini` file key
Check inactive users*	Checks for inactive logged users.	Yes, No	No	[Scheduler] UserWatchDog
User timeout*	Allowed idle time for inactive users.	00:01 - 23:59	01:00	[Scheduler] UserTimeOut

This table describes the other options:

Options	Description	Values	Default value	`ini` file key
Process ALEA-COMMAND file*	Checks the `ALEA-COMMAND` file every 5 seconds and performs the commands.	Yes, No	Yes	[Scheduler] CommandFile
Run direct importer*	Checks for the `IMPORTSTART` file every 5 seconds and performs the import.	Yes, No	No	[Scheduler] AcmeImporter
Specify SQL connection string*	Enter the parameters for the SQL connection string (see "Connection Strings" on page 22)		Server=localhost;Trusted_Connection=yes;Database=Alealog;	[Scheduler] CellToSQLConnectionString
Indicate name of SQL table*	Enter a name for the SQL table		Alealog	[Scheduler] CellToSQLTable
SQL: Overwrite Definition of the cubes*	If set to Yes, the cell changes of all cubes are transferred to SQL Server, regardless of the setting in the Definition of the cubes.	Yes, No	Yes	[Scheduler] CellToSQLOverwriteSettings

Log Settings

The Log Settings component presents settings for several protocol files.

The settings marked by an asterisk (*) can be applied to the database at runtime. Click the **Apply runtime settings** command on the shortcut menu of the database to execute the settings at runtime.

This table describes the detail level of log entries options:

Options	Description	Values	Default value	`.ini` file key
Detail level*	Detail level of the messages in the event window of OLAP Server.	Debug, Verbose, Quiet, None	Quiet	`Alea.ini` [Protocol] PrintLevel
Log level*	Detail level of messages in the `Aleapr.txt` file.	Debug, Verbose, Quiet, None	Quiet	`Alea.ini` [Protocol] ProtLevel
Detail level of EVENTLOG*	Detail level of the messages in the EVENTLOG.	Debug, Verbose, Quiet, None	None	[Protocol] EventLogLevel

This table describes the `Alealog.txt` log setting options:

Options	Description	Values	Default value	`.ini` file key
Log date*	The protocol messages include the date.	Yes, No	Yes	[Protocol] DisplayDate
Log time*	The protocol messages include the time.	Yes, No	Yes	[Protocol] DisplayTime
Log high resolution time*	The protocol messages include the time in milliseconds.	Yes, No	No	[Protocol] HighResTime
Log settings on Read	Logs the configurations used into the standard logs: `AleaPr.txt`, console window and Windows event log. The content is independent of the log level (Quite, Verbose or Debug). Passwords are replaced by the fixed string 'xxx', regardless of their value and length.	Yes, No	No	LogSettingsOnRead

Options	Description	Values	Default value	`.ini` file key
Log Version*	1: Only changed cells are logged 2: Additional information about the cell and splash operations are logged (table AlealogOperations) 3: IPv6 4: 30 dimensions 5: Includes fields for hierarchies 6: Increased string cell limit (1,000,000 bytes)	1, 2, 3, 4, 5, 6	1	LogVersion
Definitions of the cubes*	Enables logging of data changes in cubes to `Alealog.txt`. The setting Transaction only logs cubes whose transaction protocol is activated.			
Log splashing operations*	Indicates that begin, end (version 1) and parameters (version 2) of splash operations should be logged	Yes, No	No	LogSplash
Log time in UTC*	Indicates that the time stamps in the log should be written in UTC and not the local timezone of the server.	Yes, No	No	LogTimeInUTC

This table describes the `Aleapr.txt` log setting options:

Options	Description	Values	Default value	`.ini` file key
Maximum number of lines*	Maximum number of lines stored in `Aleapr.txt`.	100 - 1000000	1000	[Protocol] MaxLines

This table describes the transaction log options:

Options	Description	Values	Default value	`.ini` file key
Log level	This setting refers to the values logged in the Transaction Protocol. BASIC restricts logging to values entered manually by the user; this level corresponds to the mode used so far in Alea 4.1. ALL on the other hand records all changes made to cube values, especially those resulting from import and data area operations.	All, Basic	Basic	[LOG] Transactions

This table describes the internal settings options:

Options	Description	Values	Default value	`.ini` file key
Log files of XML requests	Enables the logging of XML requests to the database root directory.	Yes, No	No	[LOG] LogXML
Logging of requests	Enables logging of the `LOGFILE.BI`N (to be used only when recommended by the Support).	Yes, No	No	[LOG] LogFile
Logging of responses*	Enables the complete logging of request responses.	Yes, No	No	[LOG] RequestLogResponses

Event Agent Settings

The Event Agent Settings component presents settings for Event Agent.

See "Event Agent" on page 111.

This table describes the general setting options:

Options	Description	Values	Default value	INI file key
Simultaneous start of Event Agent and OLAP Server	Select YES to enable the simultaneous start of OLAP Server and the Event Agent.	Yes, No	No	[EventAgent] StartUp
Event-filter cube	Displays the active event filter cube.			[EventAgent] EventFilterCube
Enable messaging	Select Yes to enable the messaging function. Messages will be issued by utilizing the API function `SetNet Messsage()`.	Yes, No	No	[EventAgent] SendNetMessage
Send net message upon run-time error	Select Yes to issue screen messages with run-time and user-defined errors by utilizing the API function `ErrorExecute()`.	Yes, No	No	[EventAgent] SendNetMessageOnError

This table describes the log setting options:

Options	Description	Values	Default value	INI file key
Enable logging	Choose YES to enable the logging of messages in `Eapr.txt`	Yes, No	Yes	[EventAgent] PRSLogFile
Log level	The detail level of messages in `Eapr.txt`.	Quiet, Verbose, Debug	Quiet	[EventAgent] PRSProtLevel
Maximum number of log entries	Maximum number of lines stored in the `Eapr.txt`.		1000	[EventAgent] PRSMaxLogEntries

This table describes the active events options:

Options	Description	Values	Default value	INI file key
Cell change	Select Yes to enable processing of OnCellEvents by	Yes, No	Yes	[EventAgent] OnCellEvent

Options	Description	Values	Default value	INI file key
	the the Event Agent.			
Attribute change	Select Yes to enable processing of OnAttributeEvents by Event Agent.	Yes, No	Yes	[EventAgent] OnAttributeEvent
Dimension change	Select Yes to enable processing of OnDimensionEvents by the Event Agent.	Yes, No	Yes	[EventAgent] OnGenericEvent
Generic event	Select Yes to enable processing of OnGenericEvents by the Event Agent.	Yes, No	Yes	[EventAgent]
Protocol error	Select Yes to enable processing of OnProtocollErrorEvents by the Event Agent.	Yes, No	Yes	[EventAgent] OnProtocolErrorEvent

OLAP Server Communication Manager

The OLAP Server Communication Manager component provides all settings for the communication of OLAP Server Communication Managers. These settings are saved in the `Alea.ini` file.

To stop an OLAP database via a database favorite entry, you must set the Repository registration and Project Name options to the same values as specified for the OLAP database.

You can define one Repository registration. Multiple entries are not allowed. If you change these settings, you must restart of the OLAP Communication Manager.

Communication

This table describes the registration and communication options:

Options	Description	Values	Default value	`ini` file key
Communication Managers for client registration	Client asks the specified Communication Managers for list of servers.	Available Comm Managers		[CLIENTCONNECT] CM1, CM2…CMx

Options	Description	Values	Default value	`ini` file key
Communication Managers for server registration	Server registers itself at the specified Communication Managers.	Available Comm Managers		[SRVREGISTER] CM1, CM2…CMx
Communicate with other Communication Managers	Specify the OLAP Server Communication Managers which receive information from this Comm Manager.	All, Listed, None	All	[COMMEXCHANGE] SendTo
Receiving Comm Managers	List of Communication Managers to receive information from this Communication Manager.	Available Comm Managers **Note:** Enter here the computer name and not 'local'.		[COMMEXCHANGE] CM1, CM2…CMx

This table describes the Communication Manager options:

Options	Description	Values	Default value	`ini` file key
Visible	Select YES to display the Communication Manager, even if it has been started by a client.	Yes, No	No	[INIT] VISIBLE
Repository Registration	Name of the registration to be used in Repository.		Best_Practices_Templates	
Project name	Name of the project to be used in Repository.		Best Practices Templates	[COS] ProjectName

The Communication component provides options for general communication settings.

This table describes the protocol options:

Options	Description	Values	Default value	INI file parameter
Protocols				
Local	Enable local communication on this computer.	Yes, No	Yes	[COMMUNICATION] Local

This table describes the general options:

Options	Description	Values	Default value	INI file parameter
Compressing	Enable data compressing.	Yes, No	Yes	[COMMUNICATION] Pack
Encrypting	Enable data encryption.	Yes, No	No	[COMMUNICATION] Crypt
Fully qualified domain name	Select YES to use fully qualified domain names. For example, myhost.sample.com.	Yes, No	No	[COMMUNICATION] FQDN

This table describes the TCP/IP setting options:

Options	Description	Values	Default value	INI file parameter
Computer IP address	IP address of this computer. The IP address is the number assigned to the computer. This number is used for communication.	IP address in dot format		[TCPIP] TCPAddr
Listen port TCP/IP	The TCP listen port of the OLAP Server Communication Manager on this computer.	1024 - 5000	2904	[TCPIP] ListenPort
Socket buffer size	The size of the socket buffer for data transfer.	0 + MaxInt	4096	[TCPIP] SockBuffSize
Listen port UDP/IP	The listen port of this computer for the data transfer between the OLAP Server Communication Managers.	1024 - 5000	2904	[UDPIP] ListenPort

To make a database available in the network, the TCP/IP protocol must be enabled, and the Network server option must be set to Yes in the Database Settings component.

Logging

The Logging component provides the options for the `Commpr.txt` – the protocol file for the Communication Manager, located in the same directory as the executable file. These settings are saved in the `Alea.ini` file.

This table describes the detail level for log entry options:

Options	Description	Values	`.ini` file parameter
Detail level	Detail level of the messages in the event window of the Communication Manager.	Quiet, Verbose, Debug	[PROTOCOL] ProtLevel
Log level	The detail level of messages in the `Commpr.txt` file.	Quiet, Verbose, Debug	[PROTOCOL] PrintLevel
Detail level of Eventlog	Detail level of the messages in the Eventlog.	None, Quiet, Verbose, Debug	[PROTOCOL] EventLogLevel

This table describes the log setting options:

Options	Description	Values	`.ini` file parameter
Log date	The protocol messages include the date.	Yes, No	[PROTOCOL] DisplayDate
Log time	The protocol messages include the time.	Yes, No	[PROTOCOL] DisplayTime
Log high resolution time	The protocol messages include the time in milliseconds.	Yes, No	[PROTOCOL] HighResTime

This table describes the `Commpr.txt` log setting options:

Options	Description	Values	`.ini` file parameter
Path to `Commpr.txt`	The directory of `Commpr.txt`.		[PROTOCOL] Path
Maximum lines	Maximum number of lines stored in the `Commpr.txt`.	100 ... 1000000	[PROTOCOL] MaxLines

Starting/stopping Communication Manager

OLAP Server Communication Manager can be run as a program or as an NT service. Start OLAP Server as a program via **Start > Programs > Infor BI > Infor BI OLAP Server**. The ComManager is started. Individual servers are started as specified.

See "Setting up the OLAP Server" on page 69.

When running the Communication Manager as a service, no command line is given. The Communication Manager can only be started and stopped using NT Services. All settings pertaining to the database and protocol must be made using the OLAP Administration. Before executing the OLAP Server Service, some settings must be made using the OLAP Administration. The root-directory path needs to be listed correctly. For example:

```
DBROOT=C:\Users\Public\Documents\Infor\BI\OLAP\Data\
```

To make a database available in the network, the TCP/IP protocol must be enabled and the **Network server** option must be set to Yes in the Database Settings component.

Note: OLAP Server Service is installed together with OLAP Server Network Server.

Configure the start of the OLAP Server Service in the NT Services dialog box. There are three choices to start the OLAP Server Service: AUTOMATIC, MANUAL, or DISABLED.

Setting the start option for OLAP Server Service

1. Open the Windows Control Panel, double-click the **Administrative Tools** icon. Double-click **Services**.
2. The OLAP Server Service is listed in the Services window as OLAP Server logged on as Local System.

Stopping the server

1. The Communication Manager must refer to the same repository as the OLAP Server. You can set this using **Computer Configuration > Local Computer > OLAP Server > Communication Manager > Repository registration**.
2. Choose an appropriate project.
3. Start the Communication Manager and the server.
4. Connect to the server with an account that has Administer Olap Communication Manager permission on the project. By default, this is available in the MasterRole admin account.
5. Select **Stop Database**.

Increasing timeout of ComManager service (Windows 7 onwards)

If OLAP Server is running as a service and it must save a large model then the time to shutdown given to it by the operating system may not be sufficient (default value is 3 minutes). This may lead to a corrupt OLAP model. In these cases the timeout should be increased.

To increase the timeout:

1. Start Registry Editor (`regedit.exe`) and navigate to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\InforOLAPServer`.
2. On the **Edit** menu, click **New > DWORD (32-bit) Value** and enter the value name `PreshutdownTimeout`.
3. Double-click the new `PreshutdownTimeout` key and specify the new value for `PreshutdownTimeout` in milliseconds.
4. Click OK and then quit Registry Editor.
5. Shut down and then restart Windows.

Note: When setting up the System Account used to start the OLAP Server Service, make sure it has the correct access rights to the database files.

Starting or stopping the OLAP Server Service manually

1. Open the Windows Control Panel, double-click the **Administrative Tools** icon. Double-click **Services**.
2. Select **Infor BI OLAP Server**. Click **Start** to start the OLAP Server Service manually or **Stop** to stop and shutdown the Communication Manager.

Shutting down the Communication Manager

Sometimes it may be necessary to shut down the OLAP Server Communication Manager and all its servers. This can be done from the OLAP Server Communication Manager console. To do this, type `stop <Enter>` from the OLAP Server Communication Manager command line.

When the OLAP Server Communication Manager is stopped manually, the data are saved automatically.

Managing the OLAP Server

7

Utilizing multiple processors

The OLAP Server uses computers with more than one processor by allocating different requests to different processors.

Nature of SMP

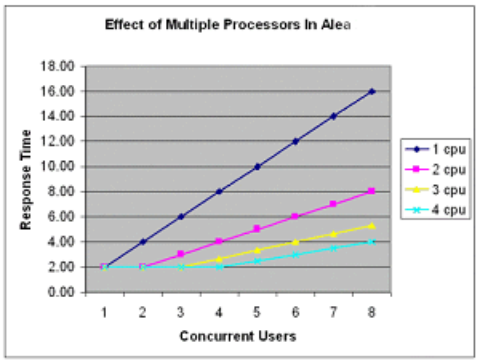
OLAP Server allocates requests from different users to the processor with the highest capacity. That means if User 1 and User 2 make a request at the same time to an OLAP Server running on a two-processor machine, OLAP Server will process the request of User 1 on the first processor and the request of User 2 on the second processor.

OLAP Server's SMP will not distribute a single request to be processed on multiple processors. Each request is always processed on a single processor.

This means that if a client is using an OLAP Server model, it will gain no speed from having a various processors. However, when multiple users are making requests to an OLAP Server, there will be significant performance gains, because the requests are processed in parallel on different processors rather than sequentially on a single processor.

Advantages and disadvantages of SMP

OLAP Server's SMP functionality only has advantages in a multi-user environment. When it is switched on, it adds some overhead to each request. This overhead may slightly slow down the processing of a request. However, in a multi-user environment, this disadvantage is more than made up for by the fact that requests from multiple users are calculated in parallel rather than sequentially, resulting in vast improvements of performance for the group as a whole. These factors must be taken into account when deciding, whether to run a model with SMP switched on or off.



Multi-core calculation (for single user)

This feature is intended for speeding up the calculation for a user on a multi-core computer if it is not fully used to its capacity. In situations where the workload is not 100%, the multi-core calculation can achieve an acceleration and reduce the response time.

In such a case the calculation is split into disjunct areas based on its dimension with the most elements. These tasks will be stored in a queue which will be processed by the worker threads. The number of worker threads is always the same as the number of processor cores. These worker threads process the tasks independently and parallel, one after another. If a worker thread is ready with its task, it takes the next task, based on the Balanced task scheduling.

Db.ini settings

Parameter	Required Format	Acceptable Values	Default Value	Description
[INIT]				
MulticoreCalc	Value	YES, NO	YES	Split a request into a number of parallel processed tasks (CSISweep).

Enabling or disabling multi-core calculation

You can enable or disable multi-core calculation. Select **Database Settings > Multi-processor settings > Multicore Calculation (for Single User)**. Then select Yes to enable or No to disable multi-core calculation.

Data model dependencies

Multi-core calculation adds some overhead to the calculation. The smaller the overhead is compared to the overall amount, the more it makes sense to pay this overhead.

Let's take a look at some situations where the OLAP Server should return/calculate some values:

- Aggregations/Consolidations

  In this scenario a multi-core calculation makes sense, as the task can be distributed with a small overhead only. This is true as long as the number of cells to add up is not too small. The OLAP Server estimates the cells involved and if they get too small, it continues with Single-core Calculation.

- A few cells calculated by rule on an aggregated level (old and new rules engine)

  Due to their small number, the cells will be calculated with single-core calculation. But if they read aggregated cells again, those might get calculated by multiple cores.

- A huge number of base level rule calculated values - new Rules Engine

  The new Rules Engine reads the parameters for the rules in big chunks. In this case the extra overhead for multi-core calculation is quite small compared to the load. Therefore we use multi-core calculation in this case. If the areas get too small, the calculation will be executed with single-core calculation.

- A huge number of base level rule calculated values - old Rules Engine

  The old Rules Engine calculates the cells one by one. It knows from the accelerator flags which cells must be calculated. Due to this the overhead for multi-core calculation would be much higher. Therefore the multi-core calculation is not implemented for this scenario. But of course, if these rules read again consolidated values, those might be calculated with multiple cores.

- Reading from Cache

  Getting values from the cache is a very fast operation in the OLAP Server. It is a single-core operation because the calculation would not benefit from using multi-core calculation in this scenario.

Issuing commands using the ALEACOMMAND file

The running OLAP Server is usually not displayed in a GUI or command line window. For example, all administration is done using the `.ini` files and Office Plus. Commands to the OLAP Server can be issued via a command file called `ALEACOMMAND`.

Using the LISTUSR function

One function, which is not readily available through the Office Plus interface, is the ability to see who is logged onto the server, or the `LISTUSR` function.

1. In order to use the `ALEACOMMAND` file the options Activate scheduler and Process ALEACOMMAND file must be enabled for the database.

2. In a directory other than the database directory, create a text file using a text editor. The file should be named `ALEACOMMAND` with no file extension.
3. Enter the command `listusr` in the first line of the file.
4. Save the file and close it.
5. Copy the file into the database directory. Within five seconds the `ALEACOMMAND` file will be deleted, because the server has executed the command: a file named `LISTUSERS` will be created.
6. Open the `LISTUSERS` file with a text editor to see the name of the user, the host server, the time the user logged in, the kernel time, the number of requests made by this user, and the date and time the file was generated.

The following line will appear in the `Aleapr.txt` file showing the command was executed successfully.

```
May/4/2004 10:25:10 ALEACOMMAND: Info >> Task: 'aleacommand' done.
```

If the `LISTUSERS` file does not appear after the `ALEACOMMAND` file is deleted, check the `Aleapr.txt` file for error messages.

To issue more than one command at a time, multiple functions may be listed, each separated by a semi-colon. All leading spaces between commands will be ignored.

The following is a list of other functions available in the `ALEACOMMAND` file:

Function	Description
`Stop`	Stops the server
`Ltab`	Loads the tables specified in the `Db.ini` into memory
`savsrv`	Saves the server
`fmem`	Clear the memory of the server
`listusr`	Lists all users logged onto the server
`rmusr`	Removes all users logged onto the server
`insch`	Starts the server's task scheduler
`rmsch`	Removes the server's task scheduler
`smp`	Dumps the present status of the SMP subsystem
`rmoneusr`	Removes one user. The ID number of the user must be specified as a parameter to this command. The ID number of the user can be determined via the listusr command
`upcos`	Reread information about roles fromRepository
`EmptyCache`	Empties the cache of the OLAP Server

Saving cubes and dimensions

OLAP Server maintains its cubes and dimensions in RAM. This means that they must be periodically saved to disk. There are three means to accomplish this. You can set parameters that will cause the server to automatically save its cubes and dimensions to disk at specified time intervals, you can use the `ALEACOMMAND` file. Or, you can manually issue a save command to the server from any client that has administrator rights.

See "Issuing commands using the ALEACOMMAND file" on page 97.

Setting automatic save parameters

You can set the auto-save parameters for an OLAP Server application using OLAP Administration. There are two settings that effect the automatic saving of the server. The first setting is First Save. This indicates the time of day when the server starts saving automatically. If this parameter is set to 15:00, the server will save at 3 p.m.

The second parameter effecting automatic saving of the server data is the Save Interval. This parameter works in conjunction with the save time. The server will automatically save at intervals (specified in hours) set in the Save interval parameter. If the save time is set to 15:00, the server will automatically save for the first time at 3 p.m. If the save interval is set to 1, it will be saved every hour from then on, or every other hour, if the save interval is set to 2.

You should normally set the automatic save to occur at a time when few or no clients access the server. It also makes sense to coordinate the save time with the daily backup routine of the file server. For example, set the OLAP Server to save at 11 p.m., when the file server is backed up at midnight. Thus the OLAP Server files are written from RAM to disk, then before new changes are made they are backed up with the file server.

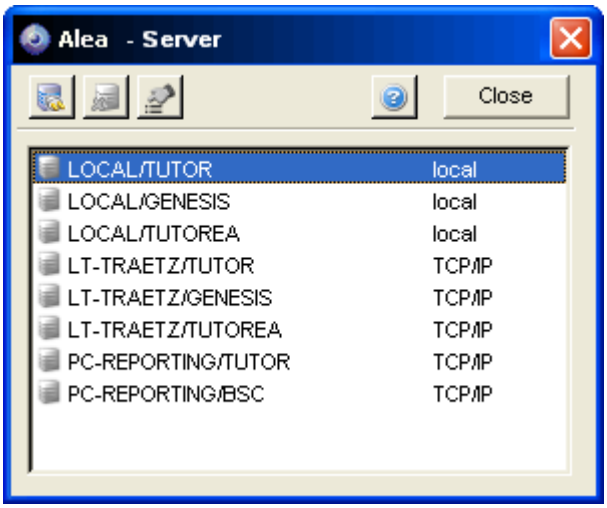
Note: To perform automatic saving, the Scheduler must run on the server.

Caution: The server is locked when saving, this means that while the files are written from RAM to disk, clients cannot make requests because the server is busy. This can play an important part in the decision for the saving interval. Saving too frequently can cause problems for clients.

Issuing a save command manually

Servers can also be saved from most OLAP Server clients. The user must have administrator rights.

1. Log on to the OLAP Server whose cubes and dimensions you want to save.
2. Select **Alea > Servers**. Or, click the **Server** button.
3. Click the **Properties** button.



4 On the Server Properties dialog box, click the **Administration** tab.

5 Click the **Save Cubes and Dimensions** button.

Activating the debug log file

The OLAP Server application server may create a special type of log file to reproduce problems, the debug log file. This file should only be activated if recommended by Infor Support. When activated, this file records each kernel request. It is written in binary format to a file named `LOGFILE.BIN` and saved to the database root directory. This file helps a programmer to review and reproduce a problem. Activating this file seriously degrades the performance of the server. It should therefore be activated only upon request of the support staff.

1 Select the Log Settings component.

2 Select Yes from the **Logging of requests** list.

To restore adequate performance of the server, set the **Logging of requests** option to No as soon as logging is no longer necessary.

Transaction logging

If desired, OLAP Server maintains a transaction log for specified cubes. This transaction log records every transaction made to a OLAP Server cube in a separate log file. This log file will be used to restore changes to a cube in the case a server was unable to save the cube before shutting down. A separate log file is maintained for each OLAP Server cube. It is named after the cube being logged and has a TRS extension. For example, TOTSALES.TRS. The file is saved in the database directory. OLAP Server LOCAL and an OLAP Server maintain the transaction log.

A transaction log record is written to disk each time a change is made to a cube for which a log is maintained. The transaction log stores the value written to the cube and the address of the cell to which the change was made. For example:

1993	Actual	Germany	ProViewV-GA12	January	Sales	100.0
1993	Actual	Germany	ProViewV-GA12	February	Sales	200.0
1993	Actual	Germany	ProViewV-GA12	March	Sales	300.0
1993	Actual	Germany	ProViewV-GA12	January	Variable Costs	400.0
1993	Actual	Germany	ProViewV-GA12	February	Variable Costs	500.0
1993	Actual	Germany	ProViewV-GA12	March	Variable Costs	600.0

Activating the transaction protocol in OLAP Administration

1. Right-click the cube you want to log from a registered and connected database and choose **Properties**. The Properties dialog opens.
2. Click the **Security** tab.
3. Select the check box **Transaction log**.
4. Click **OK**.

The transaction file is maintained as long as the corresponding cube is not saved. If you issue a cube save command, or the cube is saved automatically, the next time you change data in the cube, the transaction protocol will be cleared and restarted.

In the event of a power failure the log file is used by the server to restore the database. Upon initialization of the server, the server loads the cube that was logged. Then the server reads the appropriate log file, restoring each transaction to its value prior to the power failure. A message will appear in the `Aleapr.txt` file indicating that the transactions have been restored.

Note: If you want to log splashing operations, you must use these settings:

```
[LOG]
LoggedOperations=ALL
Transactions=ALL
```

Attribute transaction logging

Starting with Infor PM OLAP 10.1.0 attributes are part of dimensions and they are loaded and saved together with dimension elements (they are not stored in DBF files anymore).

When an attribute value is entered it is stored only in memory as long as the dimension is not saved (like cell values are only stored in memory as long as the cube is not saved).

All entered attribute values are also written to an attribute transaction log. The name of this log file is `<dimension name>.DRS`. For example, `PRODUCTS.DRS`. It will be used to restore attribute changes in the case a server was unable to save the dimension before shutting down.

The behavior of attribute transaction files is very similar to the cube's transaction files (`.TRS`):

- when a dimension is loaded the attribute values stored in the corresponding `DRS` file are loaded
- when new attribute values are entered they are immediately written to the `DRS` file
- after the dimension is successfully saved or released from memory the `DRS` file is deleted.

Importing/exporting data

Changed behavior of the cube export

Starting with OLAP Server 10.4.1 the behavior of the cube export changed:

If a cell contains a string value and the value contains a new line (`\n`), carriage return (`\r`) or tabulator (`\t`) character, these characters are escaped during export with a '^' (`^n`, `^r`, ^t).

With this change it is possible to export/import cubes with cell values containing these characters. But maybe existing export files cannot be correctly imported, if they use the '^'-character.

Direct Importer

The Direct Importer is a scheduled function that allows importing directly into a running OLAP Server without using an OLAP client. To take advantage of this import method, the flat file used must be of a specific structure. It must meet the following criteria:

1. There is one field in the records corresponding to each dimension of the cube to which you are importing.
2. The contents of the fields exactly match the elements of the dimensions to which they correspond.
3. The fields in the flat file containing the element names must be in the same order as the cube dimensions.
4. There is only one field in the records containing numbers to be imported.

5 The flat file must use the ANSI character set.

If you import very large amounts of data, it is recommended that you create such a flat file to benefit from a faster import. Also, since the Direct Importer is a scheduled function, it can be triggered automatically.

Enabling Direct Importer

1 Select t the Scheduler Settings component.

2 Select **Yes** from the **Run direct importer** list.

Once the scheduling is enabled, the OLAP Server will check every minute for the presence of the file `IMPORTSTART` (no file extension) in the database directory. The file must contain the cube name, the name of the import file, the name of the error file, and the delimiter. You may list multiple imports, as shown below. For example:

```
TOTSALES
C:\My Documents\Infor\Data\Tutor\ImportA.txt
C:\My Documents\Infor\Data\Tutor\ErrorsA.log
TOTSALES
C:\My Documents\Infor\Data\Tutor\ImportB.txt
C:\My Documents\Infor\Data\Tutor\ErrorsB.log
;
```

In the above example the two files `ImportA` and `ImportB` are imported to the TOTSALES cube. Any errors are recorded in the error log files `ErrorsA` and `ErrorsB`. The delimiter of the flat files is the semicolon. It is not possible to use the tab character as a delimiter. When the import is completed, the file `IMPORTSTART` is deleted.

Note:

- Use %#DELETED#% in the flat file to delete the corresponding value in the cube.
- When the import is finished and the specified error log file is empty, no errors occurred. Any records that could not be imported is written to the specified error log with this structure:

  Error: 'Error Number', Cube: 'Name of Cube', Elements: 'Elements1',…
- The watchdog, auto save and broadcasting functions are suspended during the import.
- It is not recommended to shut down the server during the import.
- During the import, the Direct Importer will check every 500 records for the existence of the file IMPORTSTOP. If the file is present, the import will be stopped.
- The import stops automatically, if 100 errors occur in 100 consecutive records.

Alea ad-hoc reports

When you open a cube, the initial view is always the same. You may, however, have a favorite perspective of the data to which you would like to jump immediately when you open a cube. This perspective might be your preferred starting point for navigating the cube. It is possible to save any

perspective of a cube as an Alea ad-hoc report and to open it at will, proceeding with your navigating of the cube from that perspective. An Alea ad-hoc report contains specific settings of the dimension buttons. Once an Alea ad-hoc report is opened, however, the dimension buttons can be rearranged at will.

Creating Alea ad-hoc reports

1. Start Office Plus.
2. Select **File > New Alea Ad-hoc Report**. Or, click the **New Alea Ad-hoc Report** button on the File and Database toolbar.
3. The Define Alea Ad-hoc Report dialog is displayed where you can define an Alea Ad-hoc report. Since it includes in one dialog all the functions that you perform one after the other in the Cube Browser, such as arranging the dimensions on the sheet, selecting elements for the title dimensions and the row and column headers and specifying zero suppression. Especially when you are working with large cubes using this dialog brings fast results.

   First select the Alias and the Cube for which you want to create the report. The Head section displays the title dimensions and in the Rows and Columns boxes you see the dimensions defined for the initial view. To rearrange the dimensions, position the cursor on a dimension box for instance in the title area of the dialog, hold the left mouse button down and drag the dimension to the desired position at another title position or to the Rows or Columns boxes. In this way you can rearrange all the dimensions for your report.

   To change the elements of the title dimensions or of the row or column headers, click the **Select Element** button to open the Dimension Browser and to select the required elements. To select zero suppression for the data area, click the **Zero Suppression** button and the **Key Query** button for the zero suppression in the title dimensions.

Opening Alea ad-hoc reports

Opening Alea ad-hoc reports

1. Start Office Plus.
2. Select **File > Open**. Or, click the **Open** button on the File and Database toolbar.
3. The Open report dialog displays in the left-hand pane the Global and Local Reports folder and in the right-hand pane the stored reports of each folder. Local reports are saved to the file system, whereas global reports are saved to Repository. You can specify the directory for the local reports in the Options dialog on the **General** tab. First select the corresponding folder and then open the report by double-clicking it in the right-hand pane.

Saving Alea ad-hoc reports

Saving Alea ad-hoc reports

1. Click the **Save** button on the File and Database toolbar.
2. Select the Global or Local Reports folder and enter a name for your report in the **Report name** box.
3. Click **OK** to save the report.

Converting local to global reports or vice versa

To convert a local to a global report or vice versa, you can drag and drop it to the target folder.

Changing the local reports directory

1.
2. Select **Infor BI Office Plus > Options**. Or, click the **Options** button on the File and Database toolbar.
3. Click the **General** tab.
4. In the Paths section choose the **Local folder** button to change the path to the location where your local reports are to be stored.

Flexible hosting

8

With OLAP Server Flexible Hosting a set of OLAP Server instances can be hosted on a cluster of machines under control of a single OLAP Server Communications Manager. A command-line tool (`Infor.BI.OLAP.ClusterAdmin.exe`, "AdminTool") is provided to allow administrators to start and shut down server instances on any machine in the cluster. Clients are unaware of which machine their server instance is running.

Note: OLAP Server 10.5.0 introduces a new file format for data models and converts old formats to the new one automatically. If you are using Flexible Hosting, ensure that you only copy the models to the central location after they have been converted.

Features

- Many Databases can be started with multiple nodes.
- Any ComManager can start a database on another node (the node should be part of same cluster)
- There is only one ComManager on every node.
- The databases are located in a common shared location, such as `CentralDBRoot`.

  Caution: You must use a separate `CentralDBRoot` for each Cluster.

- Each node maintains a local database folder to put log files there temporally for better performance, these files are backed up regularly on `CentralDBRoot`. Database save happens on CentralDBRoot.
- A database can be shut down from any remote Cluster ComManager, the same database can be started from other node.

Note: Currently only SMB over TCP is supported.

File location

The following files are installed with the OLAP Server setup in `C:\Program Files\Infor\BI\OLAP\bin64\`:

- `Infor.BI.OLAP.ClusterAdmin.exe` is an executable file to use on command line.
- `Cluster.ini` is a sample of the `Cluster.ini` file.

Content and location of the Cluster.ini file

The `Cluster.ini` file must be located on the network share where the databases are located.

```
[Cluster]
DBs=Demo_DB_1|Demo_DB_2|Demo_DB_3
Nodes=ClusterComp_1:2904|ClusterComp_2:2904
```

Key	Description
`DBs`	Name of the remote databases Contains a list of databases which can be started in a cluster on different nodes. If the database is configured to start in Automatic mode like <DBName:A>, it should contain the list of nodes on which it should start. For example, by <DBName:A:<Node1:Node2>>. The default mode is Manual.
`Nodes`	Contains a list of nodes which can be part of Cluster. `Nodes` also contains information about the port, but this information is not mandatory. Default port is 2904.

Settings in the Alea.ini file on the local computer

To enable the cluster feature on a local computer a `[Cluster]` section must be added to the `Alea.ini`:

Example:

```
[Cluster]
ClusterEnabled=Yes
ClusterName=DemoCluster
CentralDBRoot=\\RemoteComputer1\CentralDBRoot
[COMMEXCHANGE]
```

```
# the other nodes, rewpectively
CM1=RemoteComputer2
CM2=RemoteComputer3
SendTo=LISTED
```

In addition a `[Cluster]` section must be added to the `Db.ini`:

```
[Cluster]
ClusterEnabled=Yes
```

Note: To make the remote databases visible for the local ComManager (dump or GetServers), configure the network visibility accordingly: `[COMMEXCHANGE]/SendTo` must be set to Listed or All.

Key	Description
`ClusterEnabled`	Yes = Cluster feature is enabled
`ClusterName`	Name of the cluster
`CentralDBRoot`	Path to the remote DB directory

Commands, Parameters and Operation Modes

Command line

```
Infor.BI.OLAP.ClusterAdmin.exe [-?] | [-i ComManager[:port]] | [-s
ComManager[:port] -c "command parameters"]
```

Command	Description
`-?`	Displays basic help.
`-i`	Starts the ClusterAdmin tool in interactive mode.
`ComManager[:port]`	Specifies a ComManager or Infor BI OLAP Service that is part of the cluster The default port is 2904
`-c`	Executes a command
command parameters:	
`Information`	Provides basic information about the cluster
`ListNodes`	Lists all the nodes in the cluster
`ListDBs`	Lists all databases in the cluster with some basic information

Command	Description
`StartDB DatabaseName TargetComManager [-u username] [-p password]`	Starts a database on a node
`StopDB DatabaseName [-s TargetComManager] [-u username] [-p password]`	Stops a database
`MoveDB DatabaseName [-s SourceComManager] TargetComManager [-u username] [-p password]`	Moves a database from one node to another.

The AdminTool can operate in two different modes:

- Non-interactive mode

  All commands must be entered with a preceeding -c

  The ComManager must be explicitly specified for each request

  Example: Listing all databases in the cluster in non-interactive mode

```
Infor.BI.OLAP.ClusterAdmin.exe –s ComManager[:port] –c  “ListDBs”
```

- Interactive mode

  The tool does not end after each command. The ComManager specified by the user will be used for the entire session. Inside the interactive mode, the commands are entered without a preceeding -c.

  To start the interactive mode:

```
Infor.BI.OLAP.ClusterAdmin.exe  -i ComManager[:port]
```

  After the inital connect the tool will contact this same ComManager for the whole session.

  Example: Listing all databases in the cluster in interactive mode

```
DemoCluster>“ListDBs”
```

Event Agent

9

The multidimensional database system OLAP Server provides several methods to model database calculations. With Alea Business Rules the database designer may implement all typical calculations in a multidimensional database model. Apart from calculations that can be performed for dimensions or cubes, relations between cubes may also be modeled. Besides the current mathematical functions simple control structures, such as conditional branching, are available.

All calculations are real-time calculations, whose results are not stored in the database. The advantage of this procedure is obvious: system data are always up-to-date and data modifications can be directly evaluated. This method also prevents multidimensional data stock from 'exploding' due to the enormous number of calculations. However, there is an obvious disadvantage: since the requested data are constantly recalculated, database responses need more time.

Referring to applications for planning and simulation, real-time calculations may be considered adequate. Considering reporting systems, however, certain calculations are only calculated once, their results are stored in the database and read out upon the next query. Constant recalculation is thus unnecessary. Therefore, to optimize the response time of a database system, disabling real-time calculations may be useful.

Furthermore, model calculations in OLAP systems may be too complex to be modeled with current methods: calculation models for business planning with their multitude of dependent parameters may not be implemented at all, or only in a limited sense, with Alea Business Rules. Hence, a more powerful development environment is required.

PushRules remedy exactly these shortcomings and complement the calculation functionality of OLAP Server. Accordingly, it is the primary aim of the PushRules to store calculation results in the database. Furthermore, far more complex rules than those hitherto applied are to be implemented. For this purpose, Infor has developed Event Agent. Event Agent executes rules, developed by the database designer, which are programmed in development environments such as Microsoft Visual Basic or Microsoft Visual C++ (COM interface is necessary).

The basic principle of the PushRules is the event-driven execution of calculation rules. In the OLAP Server database any act of writing in cubes, attribute tables or dimensions is considered an event. These events are evaluated by the EventServer Connectivity. Rules will be linked to and executed within the database via event filters.

Since this manual focuses on the experienced Alea database designer, appropriate programming skills as well as thorough experience with Microsoft Visual Basic are required.

Program Components

Event Agent is installed together with the OLAP Server Network Server components (network server is necessary). The installation comprises the following software components:

Component	Description	Destination
`Mis.Alea.EventAgent.exe`	The `Mis.Alea.EventAgent.exe` file initializes the PRSEngine etc.	Program directory
`EAPR.txt`	The `EAPR.txt` file is the event log of the Event Agent. The file can be read with any ASCII editor. It is created upon the start of the program.	Database directory
`EVA.rw`	Write-protected file that only exists while Event Agent is running. With this file Alea Server checks whether Event Agent is running.	Database directory
`Db.ini`	The `Db.ini` file contains the program configuration.	Database directory
`EventSave.xml`	If non-processed events are queued when the Event Agent is stopped, these events will be saved in the database directory as `EventSave.xml`. The file will be loaded and processed with the next start of Event Agent.	Database directory
`Mis.Alea.SystemDiagnostics.dll`	C#-Assembly for performance counter	Program directory
`Mis.Alea.SystemDiagnostics.tlb`	In terms of function, it is identical to `Mis.Alea.SystemDiagnostics.tlb`. This files is necessary to communicate to a COM using a .Net assembly.	Program directory
`Mis.Alea.EventAgentEngine.dll`	The functions library `Mis.Alea.EventAgentEngine.dll` represents the core component of the Event Agent and reacts to events, compares modified database cells with the trigger tables and calls functions of the program libraries.	Program directory
`mdsESC.dll`	Event Server Connectivity, which is the part that receives the	Program directory

Component	Description	Destination
	events from OLAP Server, inserts them into a queue and transfers it in a second thread (of low priority) to Event Agent.	

Annotations

- Event Agent, which is a client application to OLAP Server, logs on to the database as user 'AleaInternal' via local.

  The AleaInternal user is created as default user by Infor BI Repository Administration.

  Caution: Do not to change the AleaInternal user.

- For the connection to Event Agent a second IP connection is established for which no additional server port is used.
- PushRules may initiate write operations that must not be carried out by the user due to his assigned user rights. This behavior does not threaten access security, but must be considered when developing PushRules.
- Since Event Agent is an OLAP Server client application, all write operations of Event Agent are also CellChange events. However, these entries are ignored to prevent circular references.

Note: Each instance of OLAP Server starts and controls its own Event Agent. You cannot start Event Agent manually or by any other method.

Event Agent is only displayed in the Task Manager on the **Processes** tab (`EventAgent.exe`). It is not displayed on the **Applications** tab.

Note: During the initialization of the Event Agent, the reservation mode of OLAP Server is active. No other user is able to connect.

When you enable the reservation mode the following message is displayed in the console window and logged in the `Aleapr.txt` file:

```
Initializing Event Agent: Reservation mode of Alea Server is active.
```

When the initialization is finished the following message is displayed in the console window and logged in the `Aleapr.txt` file:

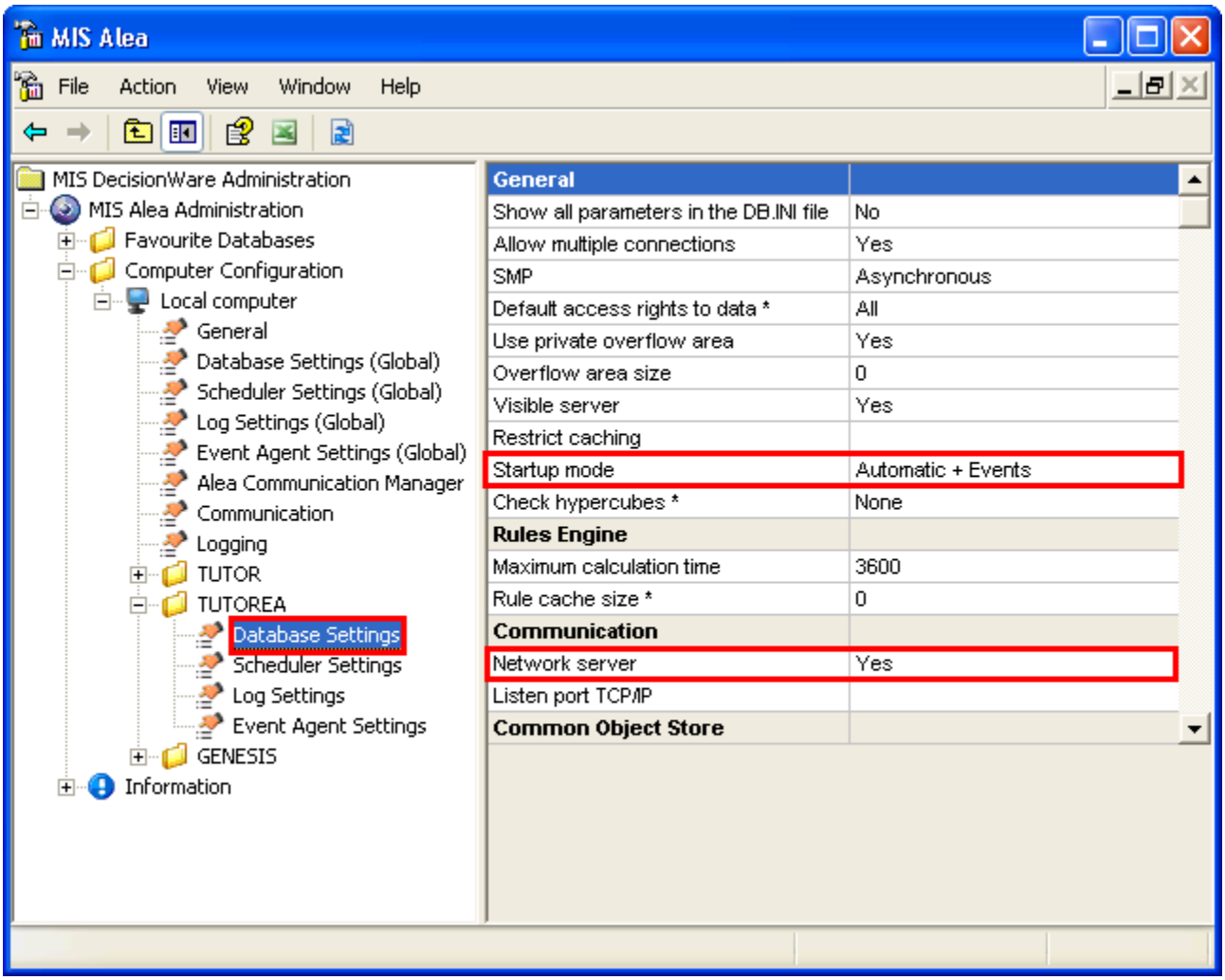
```
Event Agent is initialized. Reservation mode is no longer active.
```

Configuration

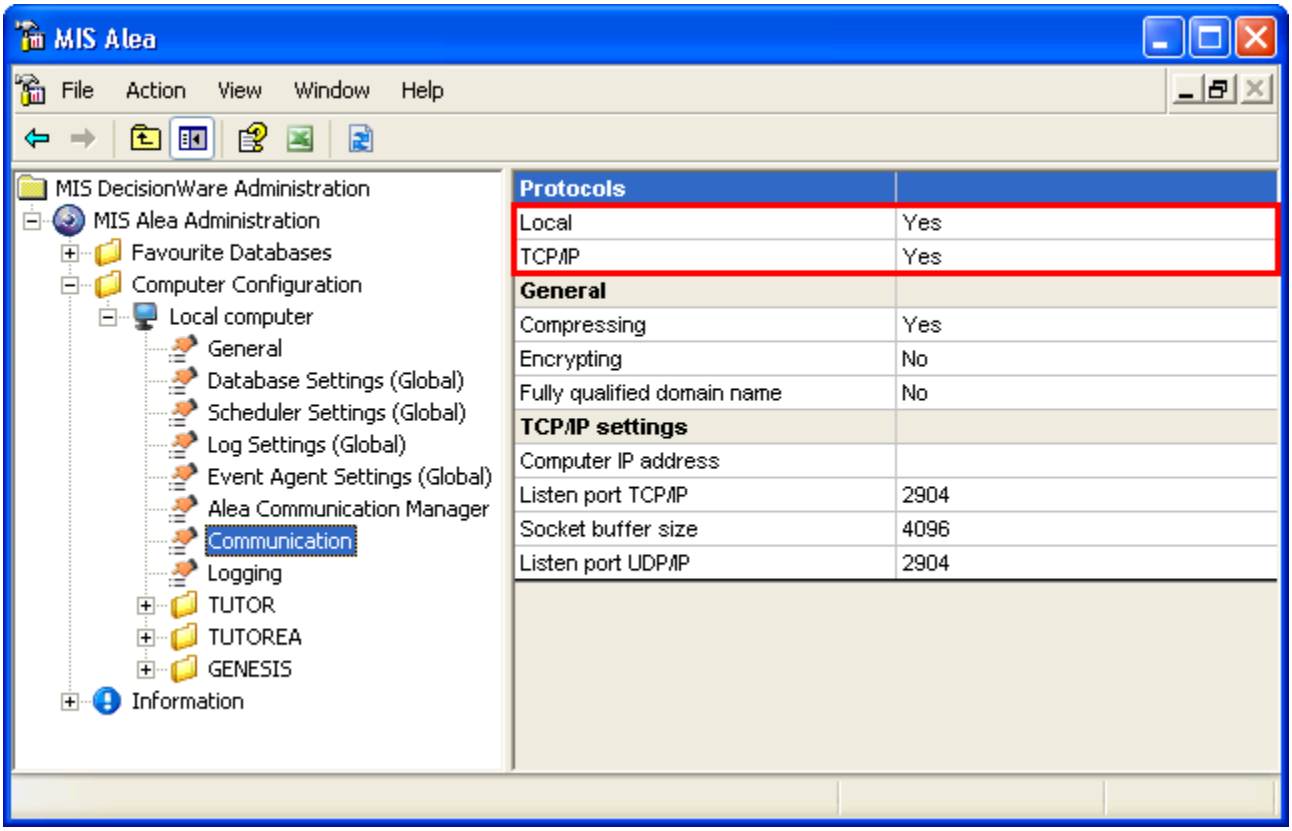
To start Event Agent and OLAP Server simultaneously, adjust the settings in the OLAP Administration. In Event Agent Settings, set Simultaneous start of Event Agent and Alea Server to Yes.

In Database Settings, set Startup Mode to Automatic + Events or Manual + Events and Network server to Yes.

Note: If Automatic + Events or Manual + Events is set to Yes, the `Db.ini` key `EnableEvents` is set to Yes.



In Communication, adjust the network settings. Make sure that both communication protocols, Local and TCP/IP, are enabled.



Event Agent Settings

Use OLAP Administration to check and modify the configuration of the Event Agent, either Global for all new databases or for an individual database.

Event Agent Settings presents settings for Event Agent. The table to the right of OLAP Administration displays the options in the left column and a lists or entry fields for the settings to the right. In the lower part of the table you find descriptions of the options (which are displayed when you click them).

General settings

Simultaneous start of Event Agent and the Alea Server

Select Yes to enable the simultaneous start of the Alea Server and the Event Agent.

Default Value: No

Event-filter cube

Displays the name of the active event filter cube. For example, STRIGGER for TUTOREA.

Enable messaging

Select Yes to enable the messaging function. Messages are issued using the API function `SendNet Message()`.

Default Value: No

Send net message upon time error

Select Yes to issue screen messages with run-time and user-defined errors using the API function `ErrorExecute()`.

Default Value: No

Log settings

Enable logging

Choose Yes to enable the logging of messages in the `Eapr.txt`.

Default Value: Yes

Log level

The detail level of messages in the `Eapr.txt`.

Default Value: Quiet

Values: Quiet, Verbose, Debug

Maximum lines

Maximum number of lines stored in the `Eapr.txt`.

The default value is 1000.

In addition, logging can be defined for each configuration in Repository Administration.

There are four types of event logging that can be used by different filters in Repository Administration:

- Text file: event information is written to a text file.

- Event Viewer: event information is forwarded to the Windows Event Viewer.
- Log Service: event information is forwarded to the Repository Administration Log Service on a remote computer.
- Custom: event information is forwarded to an external logging module.

Active events

Cell change

Select Yes to enable processing of `OnCellEvents` by Event Agent.

Default Value: Yes

Attribute change

Select Yes to enable processing of `OnAttributeEvents` by Event Agent.

Default Value: Yes

Dimension change

Select Yes to enable processing of `OnDimensionEvents` by Event Agent.

Default Value: Yes

Generic event

Select Yes to enable processing of `OnGenericEvents` by Event Agent.

Default Value: Yes

Log error

Select Yes to enable processing of `OnProtocollErrorEvents` by Event Agent.

Default Value: Yes

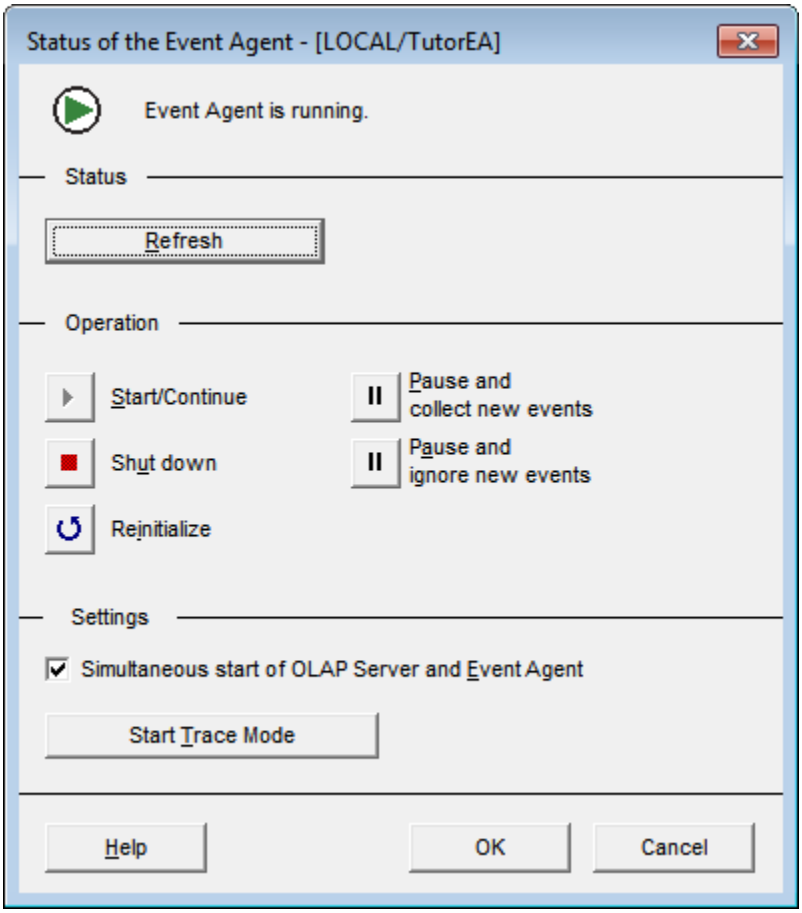
BulkImport

You can suppress the generation of events for certain users by assigning them the BulkImport role. Thus, the performance of the Event Agent is improved. For further information about assigning roles in the Repository Administration refer to the online help for Repository Administration.

Event Agent status

Open the Status of the Event Agent dialog by selecting **Status of the Event Agent** on the shortcut menu of the registered and connected database.

The Status of the Event Agent dialog allows you to control Event Agent.



To display the current status of the Event Agent, click the **Refresh** button in the Status section. The Operation section provides command buttons to start, pause, and shut down the Event Agent. Click the **Reinitialize** button to update the database after you have changed settings of Event Agent (either using OLAP Administration or the `Db.ini`) or edited Event Filters outside the Event Filter Editor.

Note: If non-processed events are queued when Event Agent is stopped, these events are saved in the database directory as `EventSave.xml`. The file will be loaded and processed with the next start of Event Agent.

To start the Event Agent and the Alea Server simultaneously, select the **Simultaneous start of Alea Server and Event Agent** check box.

The Event Trace Mode displays the events of the local server. Click the **Start Trace Mode** button. The Event Agent Trace Window dialog opens as soon as an event is triggered and shows the parameters of the triggering event and the released event filters.

This function can only be run on a local Alea server. The dialog opens for each event and must be closed by either clicking **OK** or **Stop Trace Mode**.

Event filters

Open the `Edit Event Filters` dialog by clicking **Edit Event Filters** on the shortcut menu of the registered and connected database.

This dialog box allows you to create, to structure and to administer event filters and event-filter cubes.

Overview of the dialog box

The left navigation pane displays the event filters and groups and provides administration functions. The right part offers you two modes that you can switch to clicking the **Mode** toolbar button. Edit mode allows you to display and edit the definitions of event filters, whereas list mode shows a list of the event filters.

To create event-filter cubes, click the **Create Event Filter Cube** button and enter a name for the cube in the New Event Filter Cube dialog. You can create several event-filter cubes per database, for example, for testing purposes, but only one cube will be active which is the one selected in the **Active event filter cube** list. You can also modify this parameter in the Event Agent Settings.

See "Event-filter cube" on page 116.

To create and administer event filters and groups you can use the following buttons on the tool bar:

Create group. Click this button to create groups and subgroups to structure your event filters.

Create event filter. Click this button to create an event filter. If you want to define this filter, switch to the edit mode. Moving filters by dragging them in the navigation pane. The sequence of the filters in the navigation pane determines the sequence of their execution.

Create a copy of the selected event filter. Click this button to copy event filters that are to be modified.

Rename group/event filter. Click this button to change the names of event filters or groups (or press **<F2>**).

Delete group/event filter. Click this button to delete groups and event filters.

To define event filters:

1. Create an event filter or select one in the navigation pane to specify its definition in the edit mode.
2. In the **Event** list, select the event, on which the event filter will take effect. For example, CellChange.
3. In the Component to be executed section, define the module to be executed when you enable the filter. Enter the name of the dll or the project name in the **ActiveX** box. For example, prdllTUTORPR.
4. In the **Class** box, enter the name of the class that provides the interface function PushRule. For example, prTUTORPR.
5. In the **Function** list, either select standard function (NOP) or specify a name for the `PushRule` parameter that to be passed on to the 'pushrule' method. Only the `CellChange` event requires this entry.

   See "Standard function" on page 120.
6. In the **Comments** box include comments or details.
7. In the Database section, specify details of the cube and the dimensions. In the **Cube** list, select the cube the Event Filter applies to.
8. In the **Type** list, choose the element type (N or C or *).
9. In the dimension fields limit the effect of the Event Filter within the dimensions. Choose the asterisk to include all the elements. To select single elements, click the **Browse dimension** button. To select subsets, click the **Subset dialog** button. In our example we will select the Price element of the Measures dimension, and keep all the elements in the remaining dimensions.

   If the **Use wildcards** check box for a dimension is selected, wildcards are used for element names (’?’ as a wildcard for 1 character, ’*’ for 0 or more characters).

   - Example 1: : Element name "200?"

     Use wildcards is not selected: the pushrule is applied only for the element with the name "200?"

     Use wildcards is selected: The question mark is used as wildcard, the pushrule is applied for elements named "2001", "2002", etc.
   - Example 2 : Element name "20*": the pushrule is applied to "20""200", "2010", "2008", etc.

Event types

- CellChange includes the changes of cell values.
- DimensionChange includes the modifications made to dimensions.
- AttributeChange includes the changes made to attributes of dimension elements.
- Generic Event includes the user-defined events.
- ProtocolError includes the protocol errors of the Alea Server.

Standard function

Event Agent provides the following standard function:

NOP means No Operation; it is used to prevent write access to data areas. This function may be executed with user rights, if external event filters are concerned. Since internal event filters are executed via the AleaInternal user, the user rights are no longer valid. In such cases, write access is barred with NOP. No particular rights are assigned to NOP, it is only the write operation that is revoked. For example, the written value is actually stored in the database for a short time.

Administering the Event Filters list

The list mode displays the event filters. To modify the filter definitions, double-click the filter which is then displayed in the navigation pane. Use the **Mode** toolbar button to switch to the edit mode, which allows you to change the definition.

The Event log

Event Agent records all activities and errors in the event log `Eapr.txt` stored in the database directory. The most recent entry is the last in the log.

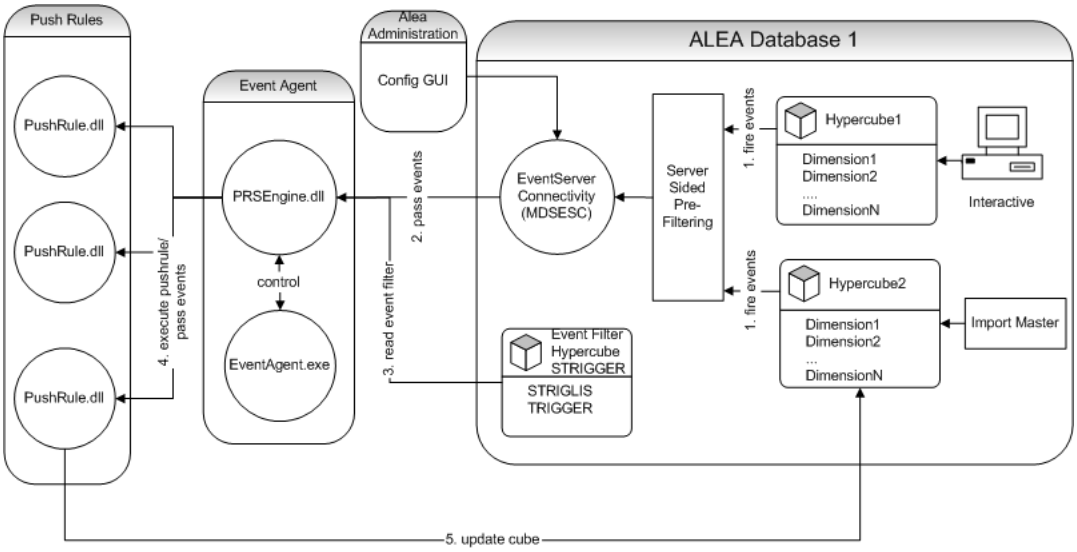
PushRules

The basic principle of PushRules is the event-driven execution of programs. Events are:

- changes of cell values by inserting, modifying or deleting cell values,
- changes of attribute values by inserting, modifying or deleting attribute values,
- modifications of dimensions by inserting, modifying or deleting dimension elements,
- generic events, to be implemented via Alea API.

These events are passed on to the Event Agent via the Alea Event Server Connectivity.

PushRules are developed as functions and stored in a function library (dll). Event Agent executes the functions on the database server. PushRule function libraries can be developed with any development environment that allow the generation of COM components. Additional libraries may be integrated with PushRules. The following figure illustrates the work mode of the Event Agent in a simplified form.



Developing PushRules with MS Visual Basic

The following seven examples illustrate the implementation of PushRules:

Note: For better comprehension of the code examples, refer to the function reference of the Event Agent API in OLAP Server VB SDK.

Step 1: Creating a project

1. Start MS Visual Basic.
2. Select **File > New Project**.

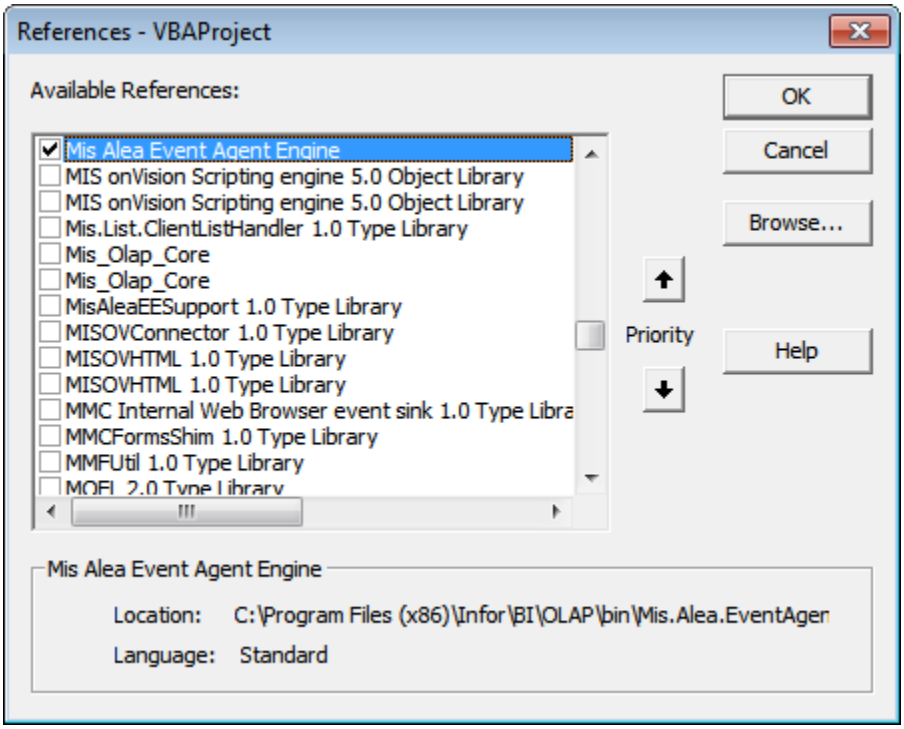
   The New Project dialog opens.

3. Choose the ActiveX DLL as the project type.

Step 2: Reference to the Event Agent API

The new VB project is open. To gain access to the API of Event Agent, create a reference to the OLAP Server Event Agent Engine (`Mis.Alea.EventAgentEngine.dll`) by selecting **Project > Reference**.

Select MIS Alea Event Agent Engine in the **Available References** list.

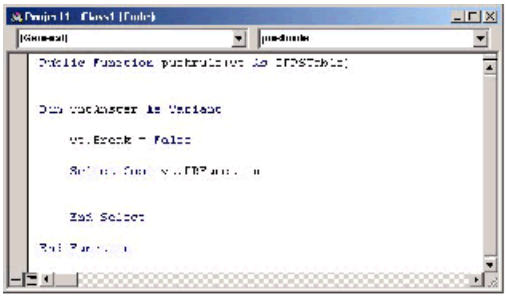


Caution: Starting with MIS Alea 5.2 the file `PRSEngine.dll` has been renamed to `Mis.Alea.EventAgentEngine.dll`.

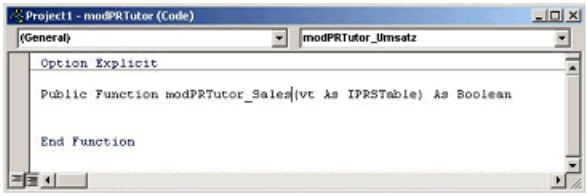
If you want to compile `dlls` developed before MIS Alea 5.2 change the reference from `MDS PRSEngine` to `MIS Alea Event Agent Engine`.

Step 3: Programming PushRule functions

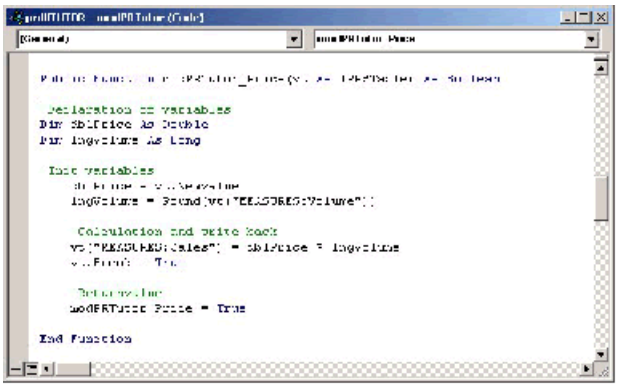
In each ActiveX VB project a class is generated, into which a public interface-function 'pushrule' for a pushrule must be implemented. The `PRSEngine` generates an instance of this class (in the example: PRSample.clsPRSample) and transfers the object IPRSTable (from the interface-class IPRSTable) as a parameter. With a Select Case control structure and the property PRFunction of the object IPRSTable you branch to the internal PushRule functions of the dll. These functions are developed either in a single or in several VB modules. See the function root below that may be used for all PushRule projects.



Create a new module by selecting **Project > Add Module** and write the function root. The following is to calculate sales, when the price is entered:

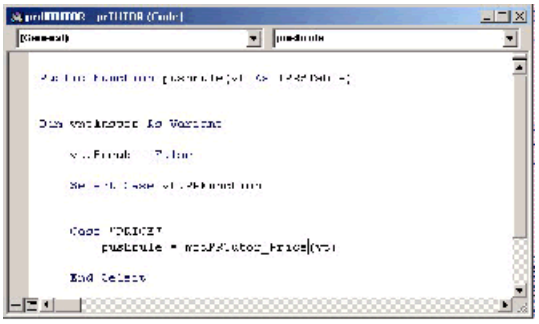


Now expand the program code. Declare the two local variables dblPrice and lngUnits, which will be initialized with the appropriate values: the entered value(vt.NewValue) will be assigned to dblPrice and the value stored in the database for the element 'Units' (apart from that, the dimensionality is identical to that of the entry). For example, vt("VALUE TYPE:Units") will be assigned to lngUnits. Subsequently, the sales are calculated as a product of dblPrice and lngUnits. The result is assigned to the vt object with vt("VALUE TYPE:Sales") = dblPrice * lngUnits and written to the Alea database. The statement vt.Break=True stops the tracking of further event filters that may have been started by the value written to sales.



Step 4: Expanding the interface function

As described in step 1, you branch to the PushRule function from the interface function using the Select Case structure. When the function 'pushrule' is called, the `PRSEngine` transfers the `PRSTable` object, and the function name of the PushRule specified in the event filter, to the property `PRSTable.PRFunction`. Expand the function pushrule so that it inserts the function `modPRTutor_Price`, when the value 'Price' is transferred as a `PRFunction`.

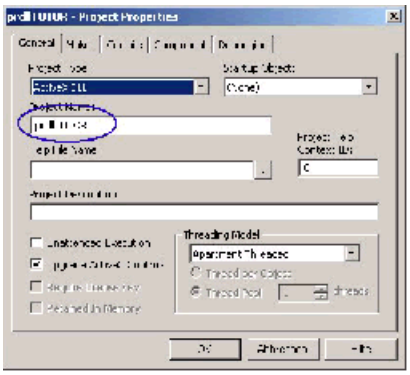


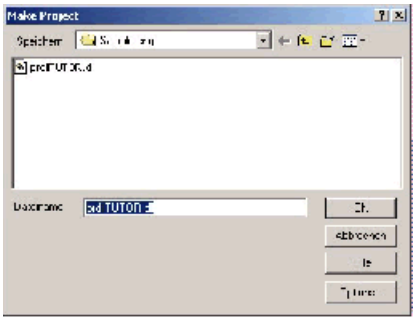
To use further PushRule functions, the Select Case structure is complemented by additional cases.

Step 5: Creating the PushRule dll

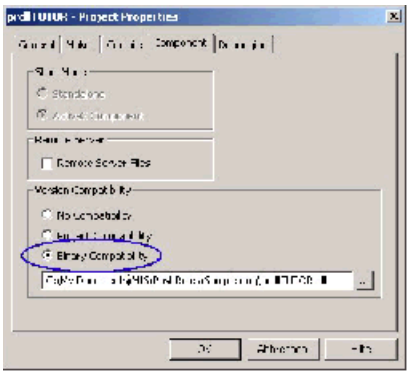
In step 5, the program is compiled and stored as a `dll` file. Store the project files first (`*.vbp`, `*.cls`, `*.bas`). Then select **File > Create 'Project Name'.dll**.

The project name of the ActiveX component and the dll name should be identical to enable debugging without changing the event filter.





Note: When the PushRule experiences any modification after this and the `dll` is re-created, the component must be binary-compatible. This prevents the `dll` from being registered with a new GUID in the system. In such a case, unregister the old `dll`, before you register the new one. Modify the project properties on the **Components** tab.



Caution: Starting with MIS Alea 5.2 the file `PRSEngine.dll` has been renamed to `Mis.Alea.EventAgentEngine.dll`. So if you want to compile `dlls` developed before MIS Alea 5.2 change the reference from `MDS PRSEngine` to `MIS Alea Event Agent Engine`.

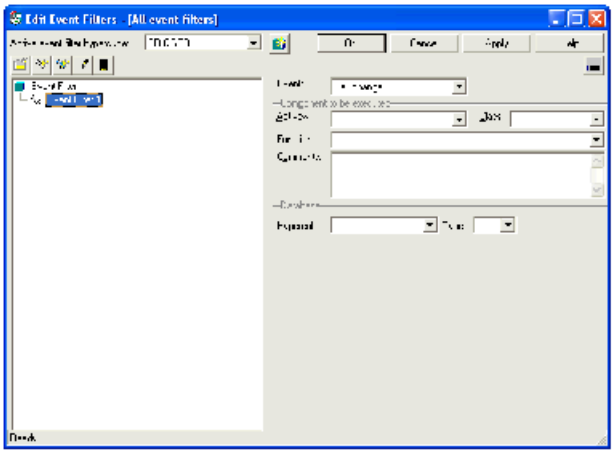
In this case you must choose the setting No compatibility for the first compilation and Binary Compatibility for all following compilations.

Step 6: Creating event filters

With the sixth step you define an event filter for the PushRule using the Edit Event Filters dialog.

Event Filter Editor

Select **Edit Event Filters** on the shortcut menu of the registered and connected database. The Edit Event Filter dialog box opens. Click the **Create Event Filter Cube** button.



Enter a name for the cube in the New Event Filter Cube dialog. for example, STRIGGER. Several event-filter cubes can be created for each database but only one event-filter cube will always be active.

The left pane of the Event Filter Editor shows the event filters as a tree view, the pane on the right displays the definition of the selected event filter. To create a new event filter, click **Create event filter** or **Create a copy of the selected event filter**. To delete an event filter, click the **Delete group/event filter** button. To keep the event filter list clearly arranged, click the **Create group** button to create new folders or subfolders and to subsume event filters under specific criteria. These folders are only used to administer the event filters and do not have any effect on their execution. To edit an event filter, select it. To display all event filters in the right pane of the editor as a list, click the **Show/hide list of event filters** button.

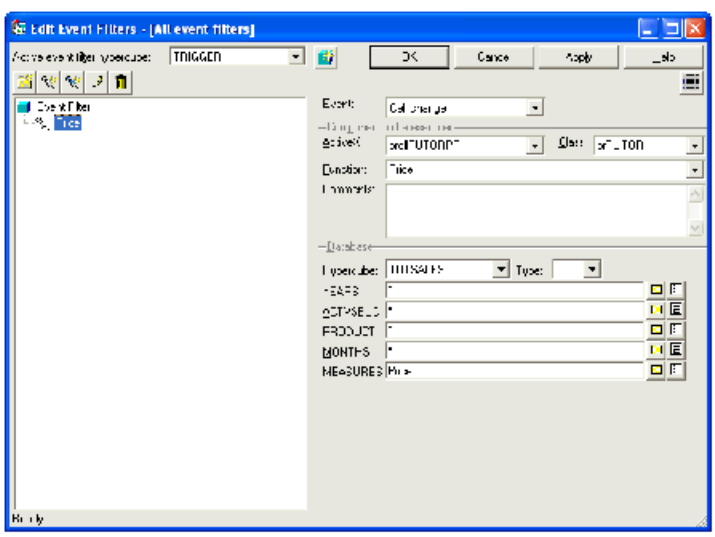
The sequence of the event filter definitions determines the sequence of their execution. If there are various event filters for a data area (identical data areas or intersections of various data areas), they are processed top down. Internal event filters, for example, event filters that initiate event filters, are taken into account.

Define and modify the event filters in the editor pane on the right. Select the event filter first to show its current properties. The meaning of the fields is presented by this table:

Event filter property	Description
Event	In the event list select the event, on which the event filter will take effect.
ActiveX	Name of the ActiveX component. For example, the project name. In tIn our example: prdllTUTORPR.
Class	Name of the class that provides the interface function pushrule. In our example: prTutor.
Function	Either select the standard function NOP from the list or specify a name for the PushRule parameter to be passed on to the 'pushrule' method. Only the CellChange event requires this entry. The entry in the example is 'Price'.
Cube	Name of the cube the event filter applies to.
Type	Type of the event filter (N or C, or N and C) depending on the type; either all elements or only leaf elements or calculated elements are triggered.
Dimension 1-n	Dimensions of the cube. The data area is defined by specifying the elements per dimension. You can choose All Elements (*), One Element, or Subset.

To store the event filter definitions in the cube, click **Apply**. Exit the Editor by clicking **OK**: the event filter list is re-initialized. If you do not want to adopt the entries, close the Editor by clicking **Cancel**.

The event filter definition for our example is illustrated by this figure:



The settings are to be read as follows: the event filter Price is defined in the TOTSALES cube. It affects the Price element in the Measures dimension. The other dimensions are not restricted (here, the event filter affects all N-elements). When an entry occurs in that data area, the method 'pushrule' with the parameter function=Price of the `prdllTUTORPR.dll` file is executed.

Standard Function

Event Agent provides one standard function that may be selected from the Function list, apart from functions developed by the user.

Function	Description
NOP	NOP means ’No Operation’. It is used to prevent write access to data areas. This function may be executed with user rights, if external event filters are concerned. Since internal event filters are executed by the user 'AleaInternal', the user rights are no longer valid. In such cases, write access is barred with NOP. Note that no particular rights are assigned to NOP, it is only the write operation that is revoked, for example, the written value is actually written to the database.

Step 7: Testing PushRules

The seventh and last step describes the function test of the PushRule. To test PushRules in the VB development environment, the project properties on the **Debugging** tab must be adapted, because a dll cannot be executed independently. Select the **Wait for components to be created** option and click **OK**.

Set breakpoints in the code lines. Start the `dll` by selecting **Run > Start** and test the source code of the PushRules within the development environment.

You may start the event filter with any OLAP Server client application by entering data. The desired PushRule will be executed and the program will be stopped at the breakpoint. Switch to Visual Basic. You will be able to analyze the source code with the usual debugger functions at runtime. Before you quit the VB debugger, quit the Event Agent in the Event Agent Status dialog (Shut down) or stop the Alea Server.

How Event filters work

Event filter types

Two different event-filter types may be distinguished:

External event filters (ExternalCall=True)	Event filters started by an external event, such as user entries, data imports, modeling actions and so on.

Internal event filters (ExternalCall=False)	Event filters started by an internal write operation. For example, those occurring in PushRules.

Execution priority

The predefined execution priority is the following:

An optional number of (external) event filters may be defined for a data area. The first external event filter (primary event filter) of a data is always of the highest priority. The priority is a direct result of the (descending) sequence of the event-filter list. When PushRules are carried out through event filters, writing values to other data areas that also include event filters (concatenated event filters), these internal event-filters are started. The next (external) event filter of the list (secondary event-filter) is only started after one event filter chain has been completely processed.

To influence the priority of execution, three commands are available in the Event Agent API:

Command	Description
PRSTable.Break	Interrupts an event-filter chain. For example, after Break=True, no further internal event-filters are traced and started. This is valid until Break=False is set. The entire event-filter list is ruled by this property. Use these commands also to prevent calculation loops between PushRules. If a circle occurs, it is reiterated for a maximum of one hundred times, before it will be resolved by Event Agent.
PRSTable.Cancel	Interrupts an event-filter list,. For example, after Cancel=True, external event-filters are no longer executed.
PRSTable.ExternalCall	Returns whether it is an external or an internal event-filter call. Use this command in a conditional branching to influence the execution sequence of the event filters.

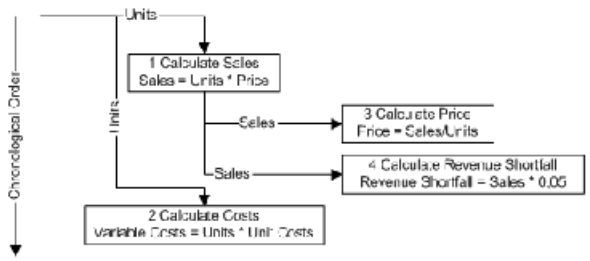
The following examples illustrate the execution of the event filters:

Example 1: concatenated event filters

We dispose of the following event-filter list:

No.	Event Filter name	Event Filter	Rule
1	Calculate Sales	Units	Sales=Units * Price
2	Calculate Costs	Units	Variable Costs=Units * Unit Costs
3	Calculate Price	Sales	Price=Sales/Units
4	Calculate Revenue Shortfall	Sales	Revenue Shortfall=Sales * 0,05

Event	Event-filter sequence
User entry 'Units'	1-3-4-2 Primary event filter: 1 Event-filter chain: 1-3-4 (primary event filter and internal list) Secondary event-filter: 2
User entry 'Sales'	3-4 Primary event-filter 3 Secondary event-filter: 4



Example 2: circle reference

We dispose of the following event-filter list:

No.	Event-filter name	Event filter	Rule
1	Calculate Price	Price	Sales=Units * Price
2	Calculate Sales	Sales	Price=Sales/Units

Event	Event-filter sequence
User entry 'Units'	1-2-1-2-1-2... Primary event-filter: 1 Event-filter chain: 1-2-1-2-1...
User entry 'Sales'	2-1-2-1-2-1… Primary event-filter 2 Event-filter chain: 2-1-2-1-2…

Resolve the circle by

1 setting the break property to 'True', before writing the value to the PushRule:

```
vt.Break = True
Sales = Units * Price
vt.Break = False
```

2 or by requesting the property ExternalCall at the beginning of both rules and by branching correspondingly. Only if ExternalCall is 'True', the rule is continued:

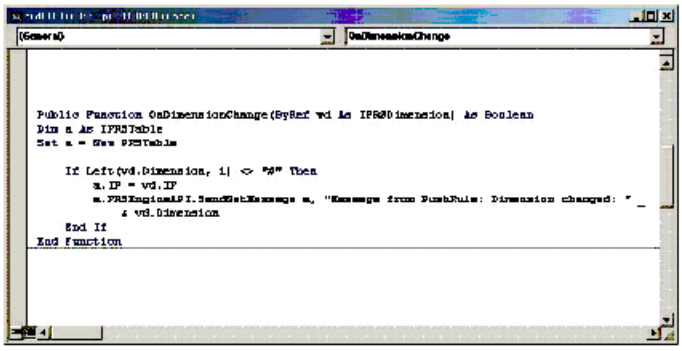
```
If vt.ExternalCall Then
Sales = Units * Price
End If
```

Event procedures

The events, dimension change, attribute change, protocol error and generic event, are handled by event procedures. This paragraph deals with the basic pattern of these procedures.

OnDimensionChange

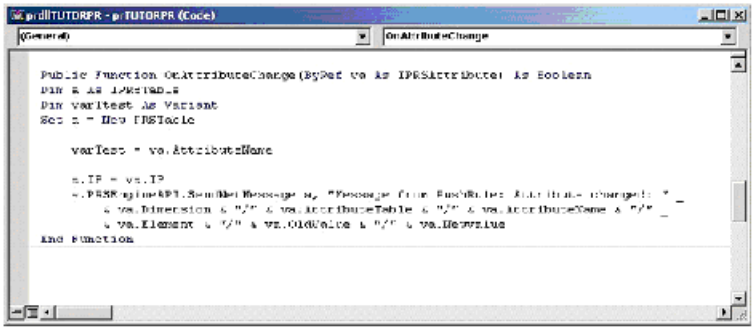
Within the Pushrule you can react to a dimension change using the event procedure OnDimensionChange. This procedure must be implemented in the PushRule class.



OnAttributeChange

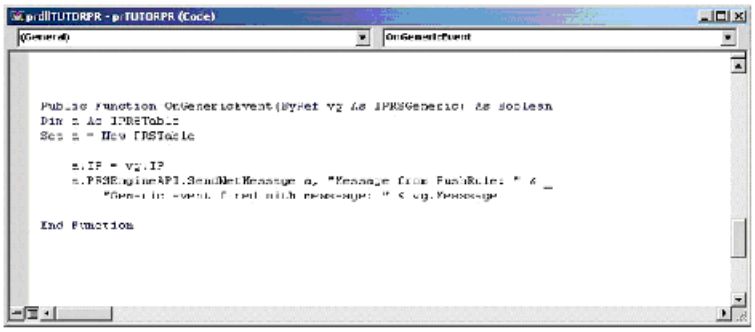
Within the Pushrule you can react to an attribute change using the event procedure OnAttributeChange. Note that this procedure is triggered for all fields of the changed attribute table. To determine the

changed values, compare the properties OldValue and NewValue. This procedure must be implemented in the PushRule class.



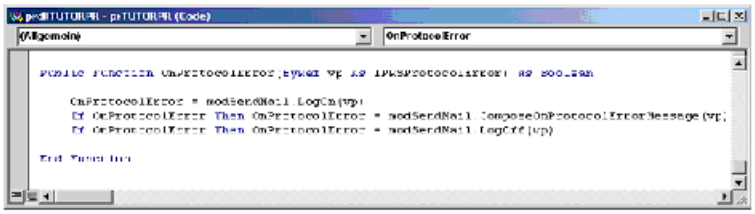
OnGenericEvent

To react to a generic event implemented by ALEA API, use the event procedure OnGenericEvent. This procedure must be implemented in the PushRule class.



OnProtocolError

Within the PushRule you can react to the error message event from an Alea database using the event procedure OnProtocolError. This event procedure must be implemented in the PushRule class. Refer to the demo PushRule for details.



If you are using the sample event handler and an error is recorded in the logfile of the server, the EventAgent raises an OnProtocolError and tries to send an e-mail.

Extended examples

The following examples increase the programming of PushRules. They explain how to implement business-administration solutions step by step.

Currency conversion

In our example below we deal with the following topics:

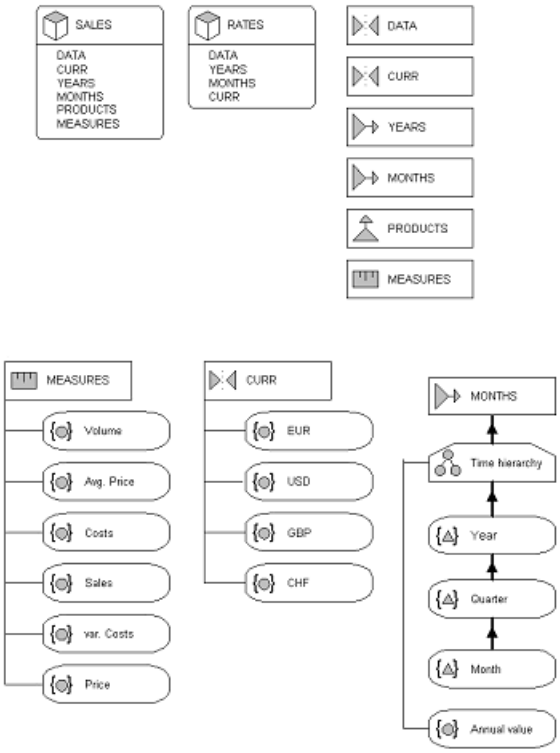
- Access to a second cube
- Using Alea API
- Different write operations of values to a cube

Analysis problem

We will convert currency values of the SALES cube using the currency rates saved in the RATES cube to the currencies of the CURR dimension. The results are displayed in the SALES cube.

Data model

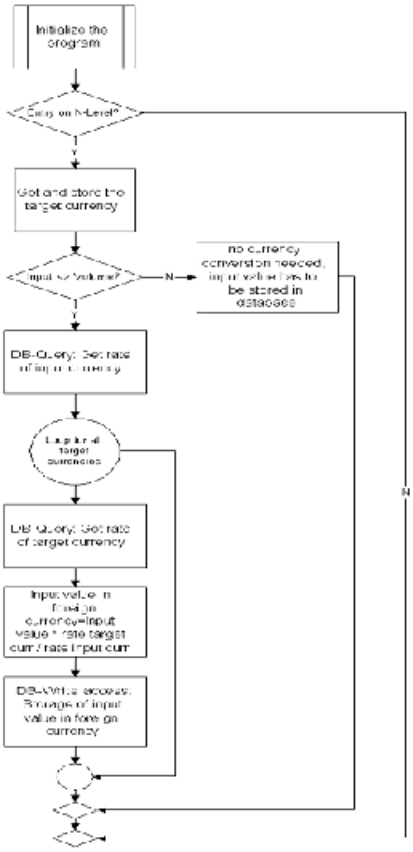
The data model consists of two cubes: SALES and RATES. The cubes contain six dimensions all together.



The MEASURES (key figures) and CURR (Currencies) dimensions are displayed in detail. Since the remaining dimensions are of little significance for this example, they are not represented in detail.

Operating concept

The following processing plan represents the program in a simplified form.



Implementation of the source code

Start the implementation of the source code by defining the program constants; then write a PushRule function named modPRTUTOR_Rates, declare and initialize all program variables:

```
Option Explicit
Public Const mcstrDIMData As String = "DATA"
Public Const mcstrDIMYears As String = "YEARS"
Public Const mcstrDIMMonths As String = "MONTHS"
Public Const mcstrDIMProducts As String = "PRODUCTS"
Public Const mcstrDIMCurr As String = "CURR"
Public Const mcstrDIMMeasures As String = "MEASURES"
Public Const mcstrCUBERates As String = "RATES"
Public Function modPRTutor_Rates(vt As IPRSTable) As Boolean
Dim vntRet As Variant
Dim dblRatesSource As Double
Dim dblRatesTarget  As Double
Dim dblValue  As Double
Dim astrCurr()   As String
Dim colCurr  As Collection
Dim vntWCurr  As Variant
```

```
vntRet = vt.AleaAPI.ElementsGetArray(vt.Server, mcstrDIMCurr, astrCurr)
Set colCurr = New Collection
Set colCurr = Nothing
End Function
```

Note: You need the collection `colCurr` to save the foreign currencies the entered values will be converted to. You can also use a simple array.

Check in the second step, whether the entry has been set onto an N-element: further processing only makes sense, if the result of this test returns 'True'. Determine the foreign currencies in a For…Each loop covering the array `astrCurr()` that was filled before with the elements of the CURR dimension. Save it in the `colCurr` collection and extend the program as follows:

```
Public Function modPRTutor_Rates(vt As IPRSTable) As Boolean
```

...

```
If vt.IsN Then
'### Determine all currencies except the entry currency
   For Each vntRet In astrCurr
      If vt.Elements(mcstrDIMCurr) <> vntRet Then
         colCurr.Add vntRet
      End If
   Next
...
End Function
```

We continue with the coding of the currency conversion. Distinguish between the entry on the element 'Volume' and the other elements of the MEASURES dimension: the volume values certainly do not require any conversion.

```
Public Function modPRTutor_Rates(vt As IPRSTable) As Boolean
...
Select Case vt.Elements(mcstrDIMMeasures)
Case "VOLUME", "Volume"
Case "PRICE", "Price", "COSTS", "Costs", "AVG. PRICE", "Avg. Price"
End Select
```

...

```
End Function
```

Develop the case 'Volume'; write the entry value for all currencies to the cube.

```
Case "VOLUME", "Volume"
'### Convert to all currencies except the entry currency
For Each vntCurr In colCurr
```

```
vt.PRSEngineAPI.PRSDATAPUTVALUE vt.Server, vt.Table, vt.User,
vt.Newvalue, _
vt.Elements(mcstrDIMData), _
vntCurr, _
vt.Elements(mcstrDIMYear), _
vt.Elements(mcstrDIMMonth), _
vt.Elements(mcstrDIMProducts), _
vt.Elements(mcstrDIMMeasures)
Next vntCurr
```

Note: Use the PRSDATAPUTVALUE command to write the values. This write access as external event, can release a event filter and additional PushRules. You may also use the vt.value command. But contrary to PRSDATAPUTVALUE this write access would be an internal event.

We continue with the coding of the other cases. Here the entry value must be converted to the target currencies. First call the currency of the entry currency from the database:

```
Case "PRICE", "COSTS", "AVG.PRICE"
'### Determine the rate of the source
vntRet = vt.AleaAPI.DataGetValue(vt.Server, mcstrCUBERate, _
vt.Elements(mcstrDIMData), _
vt.Elements(mcstrDIMYears), _
vt.Elements(mcstrDIMMonths), _
vt.Elements(mcstrDIMCurr))
```

For the conversion in a For ...Each loop using the

```
colCurr
```

collection use this formula:

(Entry value in entry currency* Rate foreign currency)/(Rate entry value)

```
'### Convert to all currencies except the entry currency
For Each vntCurr In colCurr
vt.Break = True
If vt.Newvalue <> 0 Then
vntRet = vt.AleaAPI.DataGetValue(vt.Server, mcstrCUBERate, _
vt.Elements(mcstrDIMData), _
vt.Elements(mcstrDIMYears), _
vt.Elements(mcstrDIMMonths), _
vntCurr)
If IsNull(vntRet) Then
dblRateTarget = 0
Else
dblRateTarget = vntRet
End If
If dblRateSource <> 0 Then
dblValue = vt.Newvalue * dblRateTarget / dblRateSource
Else
```

```
dblValue = 0
End If
Else
dblValue = 0
End If
If dblValue = 0 Then
```

Finally, the result is written to the database with PRSDATAPUTVALUE. Further event filters are released and the PushRules is initialized.

```
vt.PRSEngineAPI.PRSDATAPUTVALUE vt.Server, vt.Table, vt.User,Null, _
vt.Elements(mcstrDIMData), _
vntCurr, _
vt.Elements(mcstrDIMYears), _
vt.Elements(mcstrDIMMonths), _
vt.Elements(mcstrDIMProducte), _
vt.Elements(mcstrDIMMeasures)
Else
vt.PRSEngineAPI.PRSDATAPUTVALUE vt.Server, vt.Table, vt.User, dblValue,
 _
vt.Elements(mcstrDIMData), _
vntCurr, _
vt.Elements(mcstrDIMYears), _
vt.Elements(mcstrDIMMonths), _
vt.Elements(mcstrDIMProducte), _
vt.Elements(mcstrDIMMeasures)
End If
```

Next vntCurr

Modification of the Pushrule function

To call the currency conversion using an event filter, the interface function ’pushrule’ must be modified accordingly.

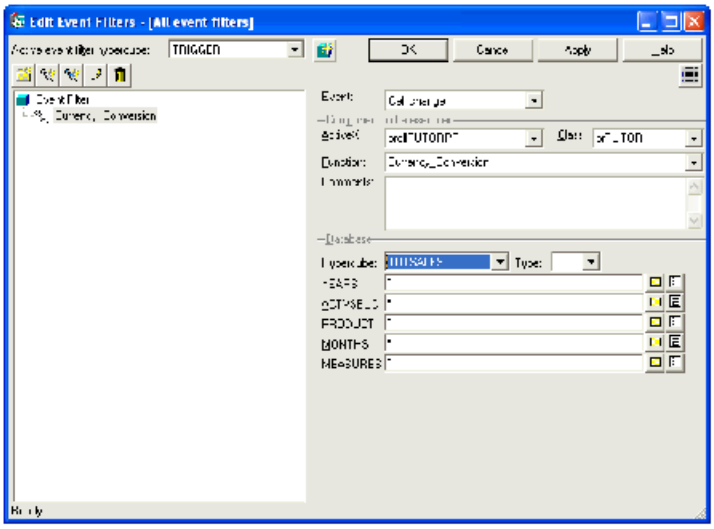
```
Public Function pushrule(vt As IPRSTable)
Dim vntAnswer As Variant
vt.Break = False
Select Case vt.PRFunction
...
Case "CURRENCY CONVERSION"
If Not gblnRunOnce Then
gblnRunOnce = True
pushrule = modPRTutor_Rate(vt)
gblnRunOnce = False
End If
...
```

```
End Select
End Function
```

To prevent the Currency Conversion function from executing a circular calculation, check, whether the currency conversion for the current entry has once been realized before, using a global variable gblRunOnce. If this is the case gblRunOnce returns the value ‘True’.

Event filter definition

The following event-filter definition applies to our example:



The Currency Conversion event-filter affects the entire TOTSALES cube. It is released on any write operation. The results of the currency-conversion release (as being events) additional event filters.

Note: To maintain the consistency of the database, another pushrule for the CURR cube must be written. When rate changes occur in the CURR cube, this rule must recalculate the values in the TOTSALES cube. Since the program operating concept is similar to the currency conversion rule, we abstain from a detailed presentation of this rule.

Distribution of entry values

Our example will deal with the following topic: Using the ALEA API (particularly DataAreaDefine, RecordLoopGetNext)

Analysis problem

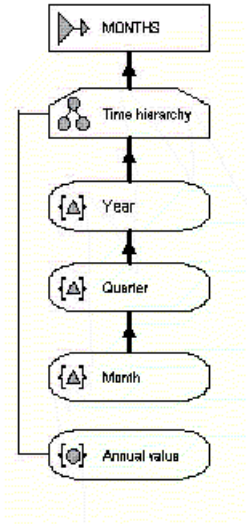
A typical application of PushRules distributes a data entry. In this example the unit price of the products is fixed for a calendar year. The prices are entered on cells of the SALES cube with the coordinates

Price (MEASURES dimension) and Annual Value (MONTH dimension). You may choose any currency, because the currency conversion implemented before calculates all currency values. To the remaining dimensions all N-elements apply. Allocate these entries to the time axis MONTHS (January to December) to use them for further sales calculations. Pay special attention to database consistency: if any prices change, all values based on these prices must be recalculated.

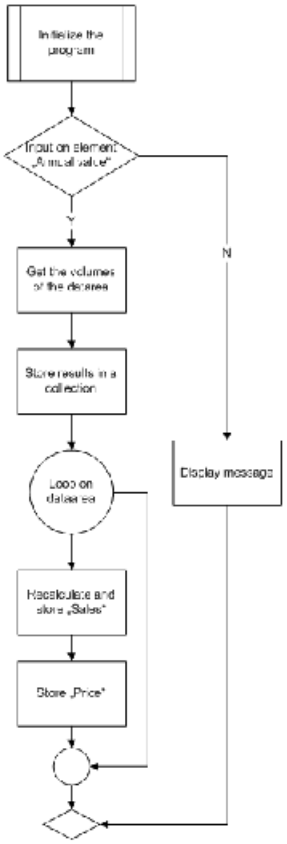
Data model

We will use the same data model as in the preceding example. The MONTHS dimension it is important for the PushRule that will be developed, we will provide a detailed explanation below.

See "Data model" on page 140.



Operating concept



Implementation of the source code

Start the implementation of the source code by defining all additional program constants; then write a PushRule function named `modPRTUTOR_Price`, and declare all program variables:

```
Option Explicit
Public Const mcstrProductMonthElement  As String = "Annual value"
Public Function modPRTutor_Price(vt As IPRSTable) As Boolean
Dim dblPrice  As Double
Dim lngVolume As Long
Dim asdefs() As String
Dim vntRet As Variant
Dim intDims As Integer
Dim intCtr  As Integer
Dim astrElement() As String
Dim colDataAreaValues As Collection
Dim vntElement As Variant
```

Check, whether the price is entered on the month element ’Year Value’ planned for this purpose. You may already complete the ’False’ branch of this conditional branching by first resetting the entry value to the old value, and by generating a notice for the user subsequently.

```
'## The price is fixed for the whole year
If vt.Elements(mcstrDIMMonths) = mcstrProductMonthsElement Then
If vt.Newvalue <> 0 Then
dblPrice = vt.Newvalue
Else
dblPrice = 0
End If
Else
vt.Cancel = True
vt("*") = vt.OldValue
vt.PRSEngineAPI.SendNetMessage vt, "For the price only entries are
permitted on the month element: " & mcstrProductMonthElement & "
End If
```

Continue with the ’True’ branch by determining the data area for that this entry was done:

```
'### Loop covering all volumes that are concerned by price changes !!!!
vntRet = vt.AleaAPI.TableDimensionCount(vt.Server, vt.Table)
intDims = vntRet
ReDim asdefs(1, intDims - 1)
For intCtr = 1 To intDims
vntRet = vt.AleaAPI.TableDimensionsName(vt.Server, vt.Table,
CLng(intCtr))
If vntRet = mcstrDIMMeasures Then
asdefs(1, intCtr - 1) = "Volume"
ElseIf vntRet = mcstrDIMMonths Then
asdefs(0, intCtr - 1) = "*"
Else
asdefs(1, intCtr - 1) = vt.Elements(intCtr)
End If
Next intCtr
vntRet = vt.AleaAPI.DataareaDefine(vt.Server, vt.Table, "", asdefs, 0,
 0, 0, 0, True, True)
vntRet = vt.AleaAPI.RecordLoopFromDataarea(vbDecimal)
```

Save the result in a collection and delete the data area.

```
Set colDataAreaValues = New Collection
Do
' get first value
vntRet = vt.AleaAPI.RecordLoopGetNext(False, astrElement())
If IsError(vntRet) Then
' If cancel condition, finish the loop
Exit Do
End If
colDataAreaValues.Add astrElement
```

```
Loop
vntRet = vt.AleaAPI.DataareaDestroy
```

Now all cells of this data area will be calculated. The sales must be recalculated with the entered price and the value for price must be stored in the database for all months.

```
For Each vntElement In colDataAreaValues
astrElement = vntElement
lngVolume = astrElement(UBound(astrElement))
vt(mcstrDIMMeasures & ":Sales", mcstrDIMMonths & ":" & astrElement(3))
 = dblPrice * lngVolume
vt.Break = True
vt(mcstrDIMMeasures & ":Price", mcstrDIMMonths & ":" & astrElement(3))
 = dblPrice
Next vntElement
Set colDataAreaValues = Nothing
```

The ’pushrule’ function is completed by assigning a return value modPRTutor_Price = True. Though the treatment of runtime errors should equally be coded, we will not consider this here.

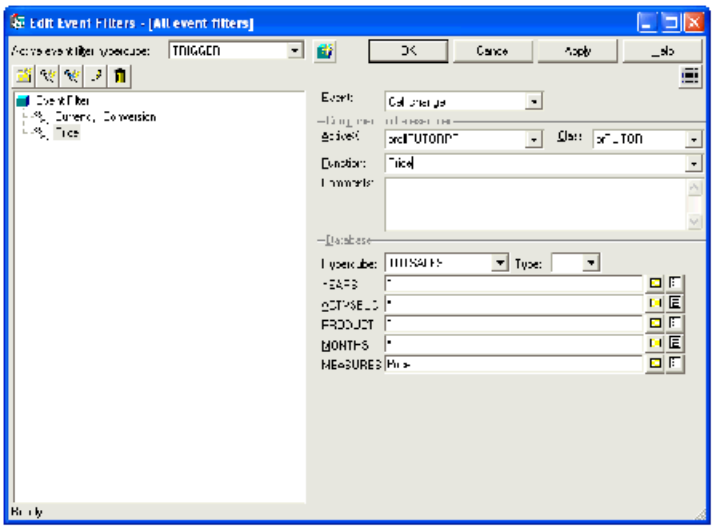
Modification of the Pushrule function

To enable the call of this ’pushrule’ function by means of an event filter, the interface function pushrule must be modified accordingly.

```
Public Function pushrule(vt As IPRSTable)
Dim vntAnswer As Variant
vt.Break = False
Select Case vt.PRFunction
Case "PRICE"
pushrule = modPRTutor_Price(vt)
...
Case "CURRENCY CONVERSION"
If Not gblnRunOnce Then
gblnRunOnce = True
pushrule = modPRTutor_Rate(vt)
gblnRunOnce = False
End If
...
End Select
End Function
```

Event filter definition

The following event filter definition applies to our example:



The event filter affects the TOTSALES cube and is released on all entries on the 'Price' element of the Measures dimension.

Changes between PushRuleServer and Event Agent

Since Event Agent 5.0 has no longer a GUI, these API methods no longer show a dialog:

- `IPRSEngineAPI.OpenSettingDialog`
- `IPRSEngineAPI.ShowEvents`
- `IPRSEngineAPI.ShowTriggerEditor`

These functions have been renamed (due to a typographical error in prior versions):

Old Name	New Name
IPRSGeneric.Meassage	IPRSGeneric.Message
IPRSProtocolError.Meassage	IPRSProtocolError.Message

FAQ

Q: Is the Event Agent 5.0 downwards-compatible. For example, is it possible to use PushRules that were developed for PushRuleServer version 1.0, 1.1, 2.0 or 2.1 in Event Agent 5.0?

A: Yes. PushRules for older versions of the PushRuleServer also run on Event Agent 5.0, if you modify the reference to the PRSEngine.dll 5.x (MDS PRSEngine 2.x→MDS PRSEngine5.x) and then compile the source code of the PushRule

See "Changes between PushRuleServer and Event Agent" on page 144 for the exceptions.

Q: Is the Event Agent compatible with Alea Server 4.1?

A: No.

Q: Why does Event Agent occupy two OLAP Server Ports?

A: Event Agent establishes two connections to the OLAP Server, one via ‘local’ for optimized access to the database, another via TCP/IP to communicate with the EventServer Connectivity.

Q: Can I replace Alea Business Rules with PushRules?

A: The answer to this question depends on various factors.The operation of both rule types is very different. Thus, they may be used in a complementary and not replaceable manner. If a decision is to be taken about which suits your problem better, always consider the entire system context.

Q: What are the mandatory elements of the Trigger dimension?

A: The mandatory elements are:

- Cube
- Type
- EventType
- Dim1 up to Dim20

The mandatory elements must be type S.

Best practices

The Event Agent may crash during large data imports (some million records). This is due to the memory restriction of the operating system of (significantly less than) 4 GB for each application process (as a 32-bit process in a 64-bit operating system). This limit may be exceeded, when large data amounts are imported in one single process and the accumulated events cannot be processed that fast (because of time-consuming PushRules). The events that are not processed yet are stored in the buffer. This will significantly increase the memory usage.

The solution is to perform the value import in separate blocks.

Accessing an OLAP Server as a client

10

Logging on to a server

Before using a database, you must log on to it, whether it is a LOCAL server or an OLAP Server. There are two ways to do this: a) log on to each server manually, b) log on to a server automatically. This results in the log on upon when the OLAP Server Excel Client interface is executed. Where a password is necessary, OLAP Server will ask for the password to the designated server with the designated user ID.

Automatic log on is also useful working with thin clients (clients with limited functionality, e.g they are unable to work on a local server). Therefore, the OLAP Server administrator can configure the client to log on to the OLAP Server automatically. The user only needs to enter his password.

To connect to a Repository dialog, click the **Log On** button at the top of the Database Structure pane. The Log on dialog opens.

The Repository is the repository of Office Plus. It contains the meta data for the user administration, report catalogs, projects and reports. To display reports in Office Plus, you must establish a connection to the Repository.

In the Log On dialog, choose the authentication system and enter the user name and the password.

If you select the **Save password** check box, the Log On dialog will not appear next time you log on. This is convenient if you regularly connect to the same repository, report catalog and project. However, if you regularly use different repositories, it may be more convenient to leave the check box unchecked.

If you have saved the password and then need to log on to a different repository you can launch the Log on dialog by logging off and then selecting the **Log On with Options** button in the Database Structure pane.

Click the **Options** button. Select the Repository you want to log on to and click **Connect**. The Project and report catalog lists are then populated with the projects and catalogs available in the selected repository. Select the project and report catalog you want to work with and click **OK**.

Disconnecting from a server

When using an OLAP Server, it is best to disconnect it when finished. Thus other users may log on to it in case of a shortage of ports on the server. After a specific period of time the server will automatically log off users who do not make a request (if the **Check inactive users** option is activated).

To disconnect the server click the **Log off** button in the Database Structure pane.

Changing the password of the ADMIN user

When you first create a OLAP Server database, it has no access restrictions. The database is created with one standard user called ADMIN. This user has administrator rights to the database. Whenever you connect to a OLAP Server database, OLAP Server first tries to log on with the ADMIN user ID. If this standard user still exists as a registered user in the database, any user will be able to log onto the database. Therefore, when restricting access to an Alea database, the first thing you must do is create a new user with administrator rights (for security reasons) and then change the password of the standard Alea ADMIN user.

1. Right-click a registered database and select **Change Password**.
2. In the Change password dialog box, first specify your current password and then your new password.
3. Confirm the change by clicking **OK**.

Passwords are case sensitive.

Encrypted communication between client and server

The new encryption of the communication in Alea uses standard algorithms for the generation of random key pairs, the generation of a session key and for data encryption. Microsoft's Crypto API already provides the basic interface to support an encryption of this type. API supports the most common data-encryption method. We generate key pairs that guarantee a safe interchange of session keys between client and server.

The public part of the key pair is transferred to the client. Then the client generates a session key for subsequent data encryption. This key is encrypted with the public key of the server. It is sent to the server which decodes the session key with its private part of the key pair. Now server and client can communicate in a secure way via this session key which only they know.

Since only the server knows the private key that is mathematically related to the public key, only the server can decrypt the session key.

Instead of using the public key directly as encryption, a session key (symmetrical key) is used for transmission in combination with the key pair (asymmetric, (public/private key)). Data encryption in combination with public encryption and private decryption would take to long, because key couples require a higher mathematical expenditure and an extended key length.

We use a DES (Data Encryption Standard) algorithm to generate session keys, and an RSA (Rivest Shamir Adelman) algorithm to generate key pairs/code pairs. Thus the length of the session key is 7 bytes. It is generated randomly per client that logs on to a database.

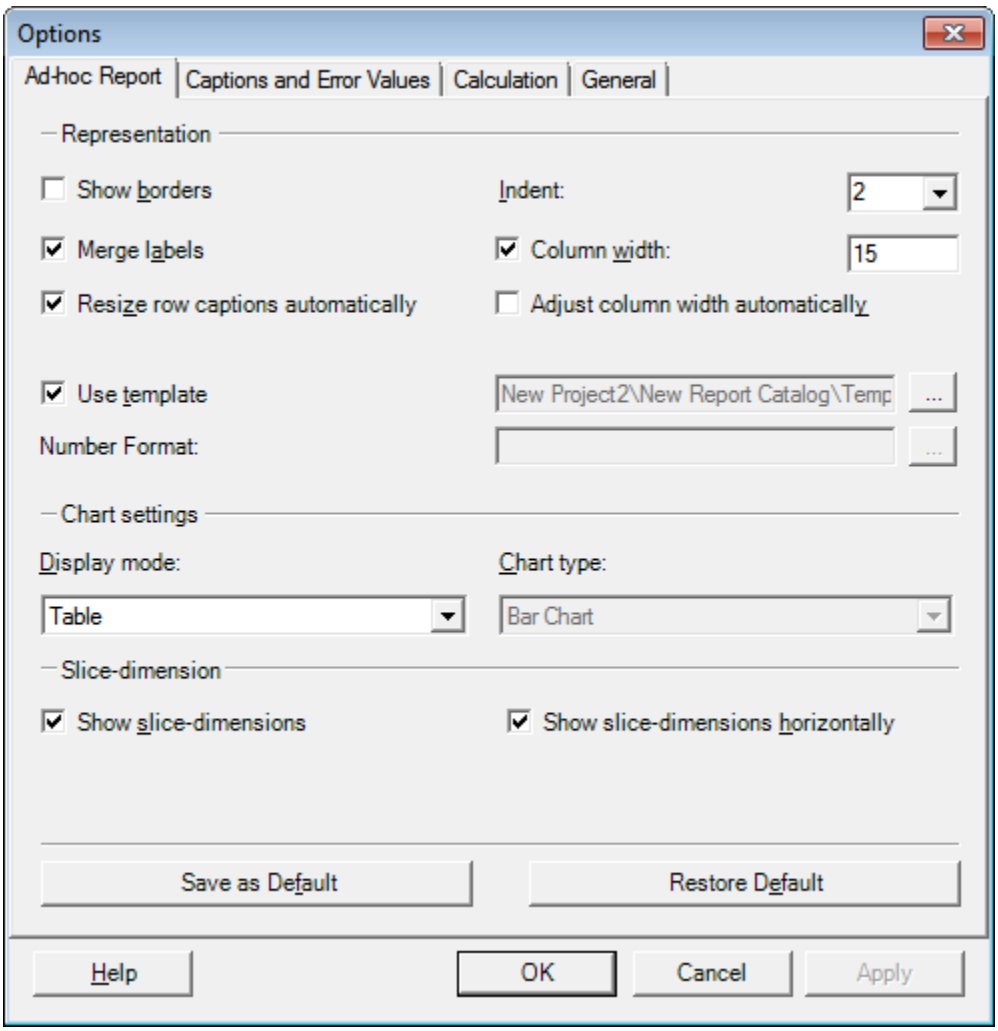
The server randomly generates the key pair with a length of 64 Bytes.

Options for Office Plus

11

This chapter explains the modification of settings in Office Plus. While working with Office Plus, users can customize the look of the dialogs: the way data are viewed, where files are saved by default and so on. The Options dialog has of five tabs, each belonging to a set of parameters which you select to customize the Excel Client. To open the Options dialog for Office Plus, click the **Options** button on the OLAP Server tool bar, or select **Infor BI Office Plus > Options**.

Report/Alea Ad-hoc Report tab

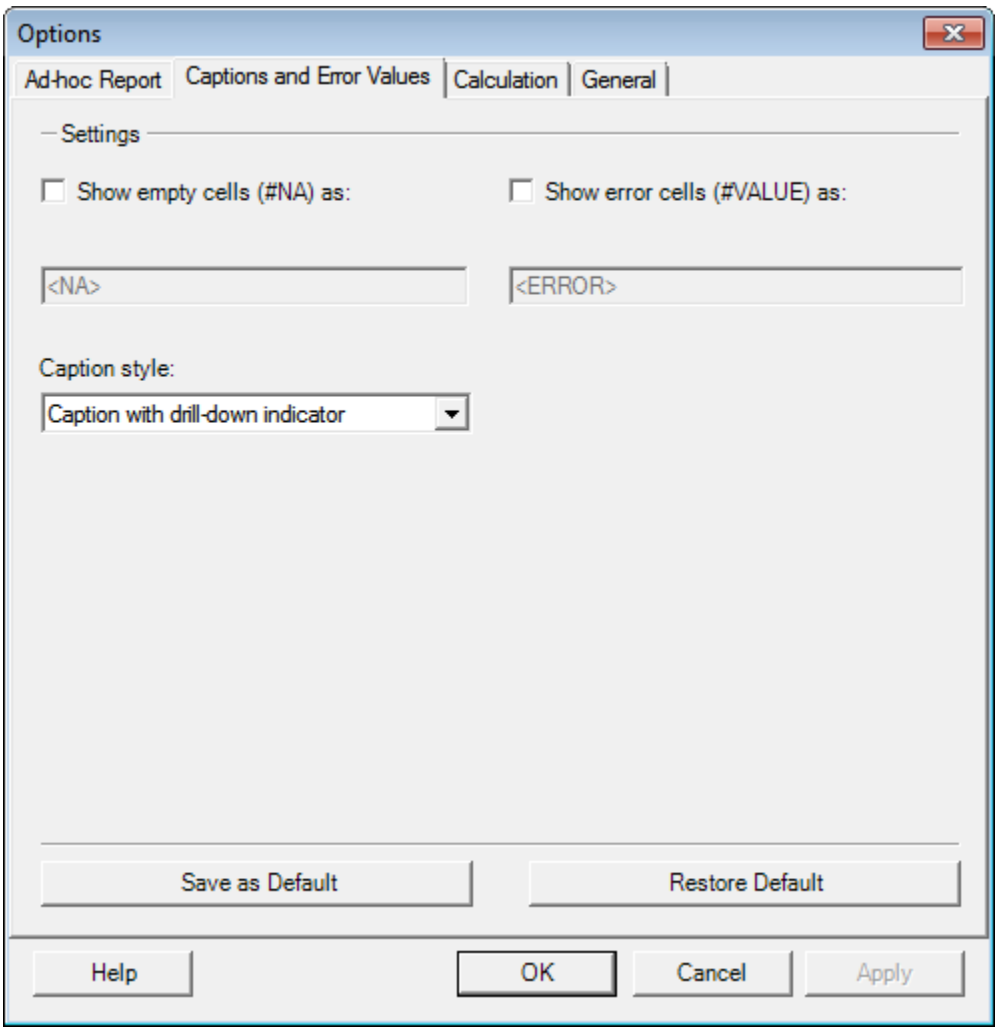


If no report is open or if an ad-hoc report is open, the tab is labeled **Report**. If an Alea ad-hoc report is open, it is labeled **Alea Ad-hoc Report**. For views, the last four of the following settings are disabled.

Setting	Description
Representation	
Show borders	Draws a frame around the line and column elements.
Merge Labels	Displays the member names over the complete width of the adjacent dimensions or lists.
Resize row captions automatically	Adjusts the width of the member names in the rows to the required width.
Indent	The selected value specifies the indent of the member names of the levels.

Setting	Description
Column width	Here you can specify a fixed column width. The check box is only active, if **Fit column width automatically** is not selected.
Fit column width automatically	Adjusts the columns width to the width of the values.
Use Template	If you select this check box, the template is used that has been selected in the text box underneath. To choose a different template, click the button.
Number format	This option is only available if **Use Template** is not selected. It opens the **Number** tab of the Format Cells dialog. Specify the number format to be used in cells. For example, date, currency and so on.
Chart settings	
Display mode	Here you specify, whether a chart is used and whether it is displayed before or after the ad-hoc report. These options are not available under Infor BI Applications Provider database aliases.
Chart type	When you use a chart, you can choose the chart type to be displayed. You cannot add a chart to an Excel report, if you use the conversion formulas CELL.GETF or CELL.GETFC (reason: the server transfers the data as strings).
Slicers	
Show slicers	Shows the slice elements in ad-hoc report
Show slicers horizontally	Shows the slice elements across the top of an ad-hoc report instead of vertically.
Save As Default	You can make changes to the report settings and save them as the default for new reports. The new settings affect ad-hoc reports and(as far as they apply) Views (Views do not support all the same settings as ad-hoc reports).
Restore Default	If you have created new default settings, and then make further changes, you can revert to your new default settings by clicking this button.

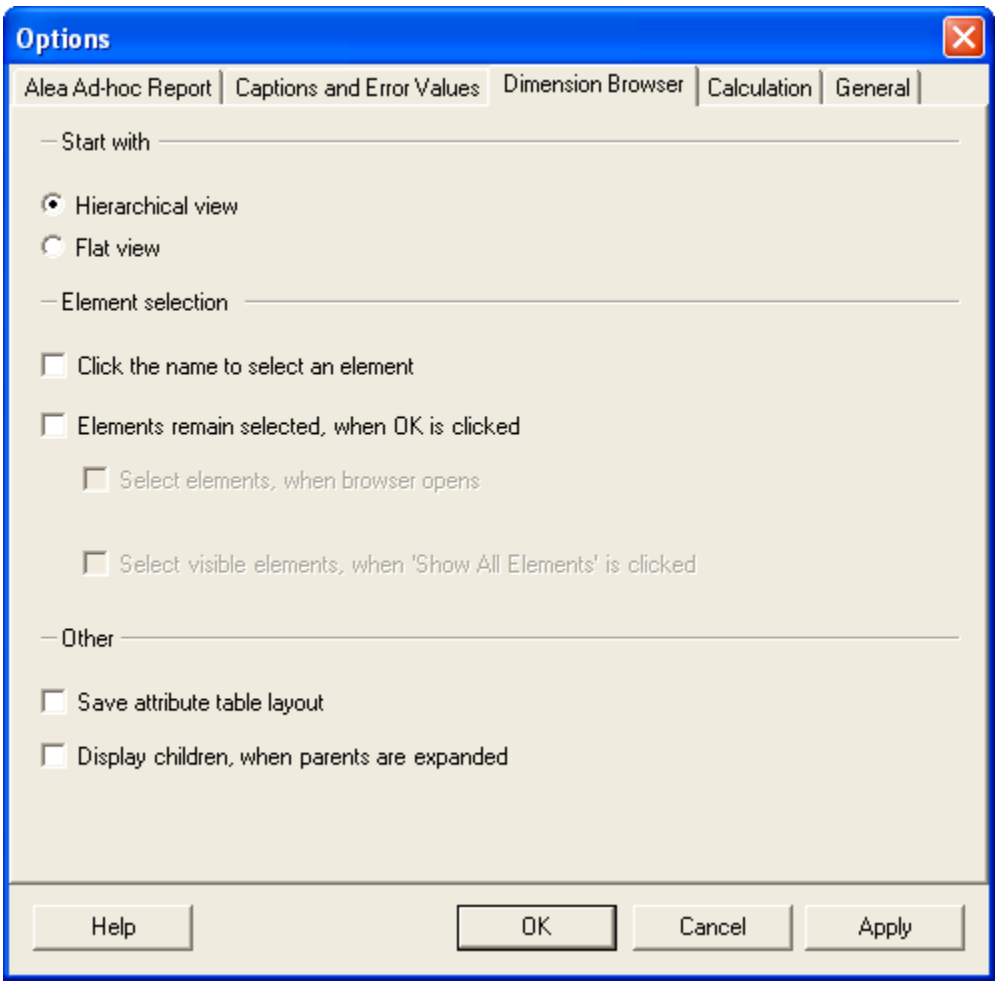
Captions and Error Values tab



Setting	Description
Settings	
Show empty cells (#NA) as	By default, <NA> is used in reports to represent empty cells. Select this check box to specify a different representation for empty cells.
Show error cells (#VALUE) as	By default, <ERROR> is used to indicate cells containing an error. Select this check box to enter a different representation for error cells.
Caption style	Here you determine how the names of elements are displayed in the report and whether the plus and minus signs that indicate the availability of drill-down are displayed.

Setting	Description
Save As Default	If you click this button, the settings of this tab are used to create ad-hoc reports.
Restore Default	If you have created new default settings, and then make further changes, you can revert to your new default settings by clicking this button.

Dimension Browser tab



On the **Dimension Browser** tab you can select various options for the display of the dimension's structure, its elements and attribute tables in the dimension browser and for the selection of elements. The options are under three headings: Start with, Element selection, and Other.

Start with

In this section you can determine the display of the dimension structure. The default setting is **Hierarchical view**. In this view, only the top level element of the dimension is displayed. You can then expand the hierarchy to reveal elements at other levels by drilling down. Alternatively, select the **Flat view** option to list all the elements, without their hierarchical structure.

Element selection

In a OLAP Ad-hoc report, you select elements for display by marking them with a tick in the Dimension Browser. By default, you then need to specify that the marked elements should be kept.

The following options are available:

- **Click the name to select an element**. Select this check box to mark elements by clicking their names. If this check box is not selected, you have to click to the left of an element in order to mark it.
- **Elements remain selected, when OK is clicked**. By default, if you have selected elements for display, you must then click the **Keep Marked Elements** button before clicking **OK** to close the Dimension Browser. By checking this check box, marked elements are automatically kept when you close the browser. The following two functions for the Dimension Browser work only in conjunction with this function:
- **Select elements, when browser opens**. This option can be useful if you are working with a large selection of elements which you want to modify only slightly. For example, you just want to remove or add a few elements. It is important to note that only visible elements are marked. The children of consolidated elements are not marked unless you have first expanded the consolidated element in the browser.
- **Select visible elements, when 'Show All Elements' is clicked**. The Dimension Browser contains a **Show All Elements** button which fully expands a hierarchy to display all elements. This option also marks all the elements.

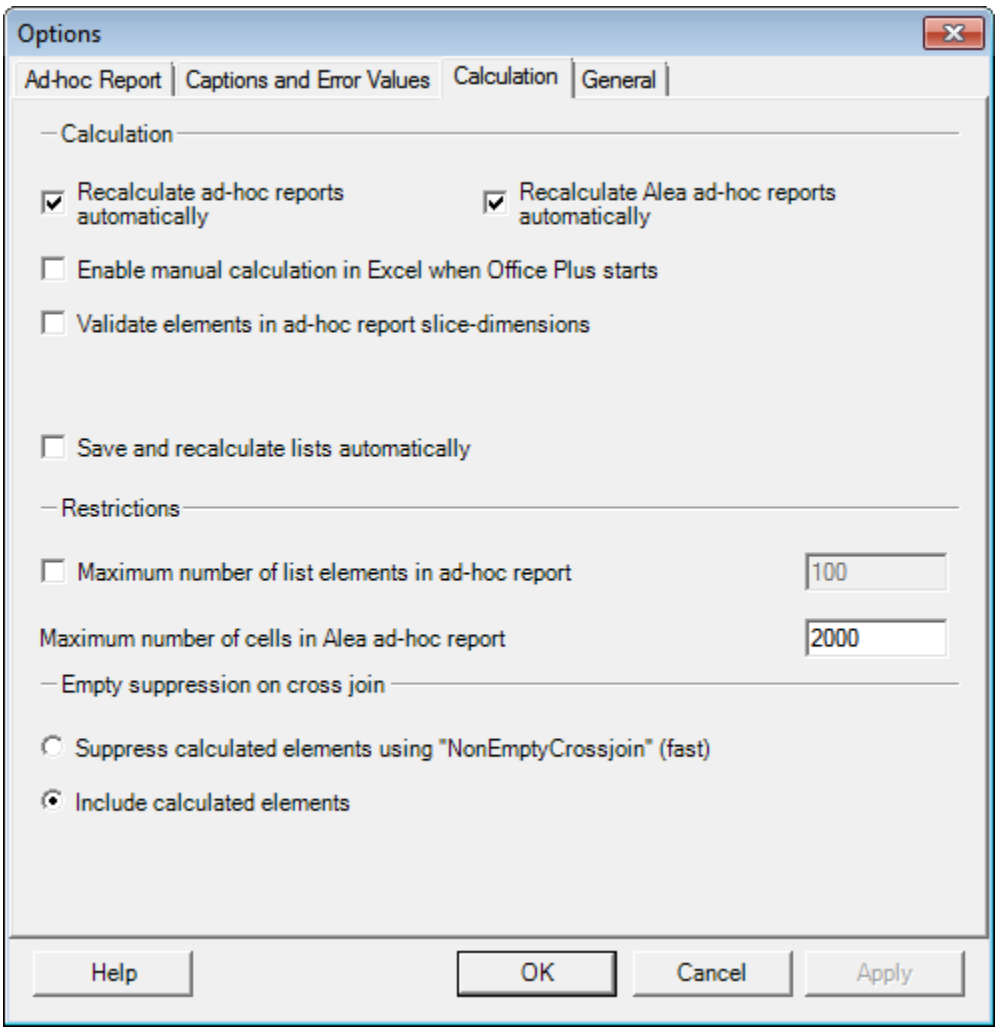
Other

A dimension may have a number of attributes associated with it. You can specify which, if any, of the attributes appear in a View and the order in which they appear in the dimension browser. By default, these settings apply only to the current report. However, you can save the settings so that the same arrangement of attributes is used in other reports containing the dimension. You do this by clicking the **Save attribute table layout** button in the dimension browser. However, this button must first be enabled here, by checking the **Save attribute table layout** check box.

For example, in a dimension containing products, you might store the product manager for each product as an attribute. In the dimension browser you could specify that the product manager attribute be displayed in a View instead of the product names. You would use the **Save layout of attribute tables** option if you wanted to apply this to all reports containing the Products dimension.

Display children, when parents are expanded. If you select this check box, the display of children will be adjusted every time you expand a consolidated element in such a way that all of the children are displayed.

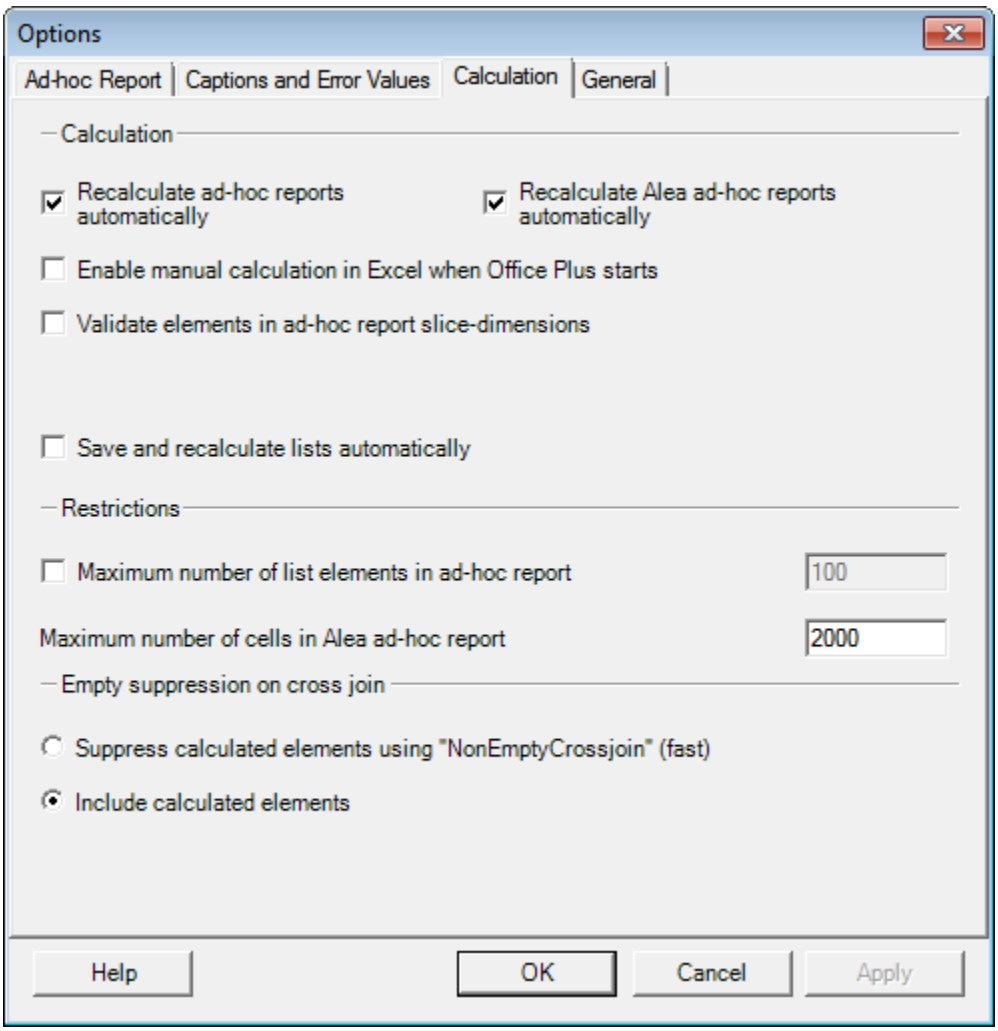
Calculation tab



Setting	Description
Calculation	
Enable manual calculation in Excel	This option turns off automatic calculation in Excel, requiring user to click **<F9>** to update Excel reports.
Recalculate ad-hoc reports automatically.	Changes to ad-hoc reports are calculated automatically. If this check box is cleared, press **<F5>** to recalculate the report.
Recalculate lists automatically	When working with the List Designer in ad-hoc reports, changes made to the definition of local lists are calculated automatically. If you un-check this box, the **Apply changes to the list** button becomes enabled in the List Designer. Click it to save

Setting	Description
	changes to the definition of the list. This setting does not affect global lists. When you edit a global list in the Database Structure pane, you must apply your changes with the button.
Recalculate Alea ad-hoc reports automatically	
Restrictions	
Maximum number of cells in Alea ad-hoc report	The greater the number of cells and calculations in a View, the longer it takes to calculate. If a view contains more than 2000 cells, a message appears, warning that calculation of the view may take some time. You can set a higher or lower threshold for the appearance of this warning.
Maximum number of list elements in ad-hoc report	By default, a maximum of 100 list elements are displayed. You can increase or decrease this figure here.
Null suppression on crossjoin	
Suppress calculated elements using "NonEmpty-Crossjoin" (fast)	The MDX-function `NonEmptyCrossjoin` is used in Null Suppression. Its effect is that null suppression is executed quickly, but calculated elements are automatically filtered out of the view.
Include calculated elements	Using the MDX functions `Filter`, `Crossjoin` and `IsEmpty` in null suppression looks for null values in all elements, including calculated elements. Calculated elements are not filtered out of the view. If you choose this option, null suppression may take some time.

General tab



The **General** tab shows the paths to various files and folders. Some are set by default, but can be changed.

Setting	Description
Paths	
Local folder	The path to the Local Reports folder in the Report Catalog.
Template folder	The path to the location where templates are stored.
Import folder	The location where files to be imported are stored
Export folder	The location where exported files will be stored.
Log file	The path to a file where errors are logged.
Ad-hoc report creation	

Setting	Description
Use all dimensions as slicers	If you select this check box, all dimensions that are not used in the rows or columns are used as slice dimensions.
User interface	
Language	Here you choose the language of the user interface (changes take effect, when you restart Office Plus). The report names, names of calculations and lists are displayed in this language, if a translation is available.

Setting up security

12

Basic concepts

OLAP Server provides extensive user security. Not only can you restrict users from accessing specific cubes within a database, you may restrict a user from seeing particular elements within a specified dimension and a specified element, or you may prevent a user from modifying dimensions and/or cube rules. The access rights to an OLAP Server application are always the same, regardless of whether you address the application locally on a standalone PC, or in a client/server environment on an OLAP Server.

Types of access rights

There are two types of access rights available for OLAP Server Databases:

- General Access Rights limit the user's ability to manipulate the database structure.
- Data Access Rights limit the user's ability to see and change data in OLAP Server cubes.

General access rights

General access rights limit the user's ability to manipulate the database structure. To assign such rights to users you must use Repository Administration.

Data access rights

Caution: If a user has the permission Administer OLAP Server Database assigned in the OLAP Permission Management, no data access rights are checked.

Access rights are assigned via OLAP Server administration cubes. These are specialized two-dimensional OLAP Server cubes that map the access rights of roles to OLAP Server cubes or elements within OLAP Server dimensions. There are three types of administration cubes.

- Cube Access Control Cube (#_TABACC cube)

- Dimension Access Control (DAC)
- Multidimensional Access Control (Multi-DAC)

OLAP Server provides a detailed security system for restricting access to data in OLAP Server cubes. You may assign READ, WRITE, NONE, ADMINISTRATOR, or DEFAULT access to whole OLAP Server cubes or to specific areas within a OLAP Server cube.

READ

READ access rights allow a user to view cubes and/or data within cubes, but does not allow a user to change data within that cube.

WRITE

WRITE access rights allow a user to view cubes and/or data within cubes and to change that data. For example, change values.

Note: For a user or a role to be assigned WRITE access to a OLAP Server cube, they must also have WRITE rights assigned under their General Access Rights. You must use Repository Administration to assign general access rights to users and roles. Refer to the online help of Repository Administration for details about the role concept.

See "General access rights" on page 161.

NONE

If a user is assigned NONE access rights to a specific cube, they will not see that cube in any list of available cubes. If they try to access the cube from a worksheet, OLAP Server will not return any values. If a user is assigned NONE access rights to specific elements within a dimension, they will not see those elements in any list of available elements. If they try to use these elements as arguments in a DBGET or DBGETC formula, the formula will not return a value.

ADMINISTRATOR

This access right allows a user full rights to the database. This can be important for example in a large enterprise in which each department has an additional administrator who monitors the applications that belong to each department.

DEFAULT

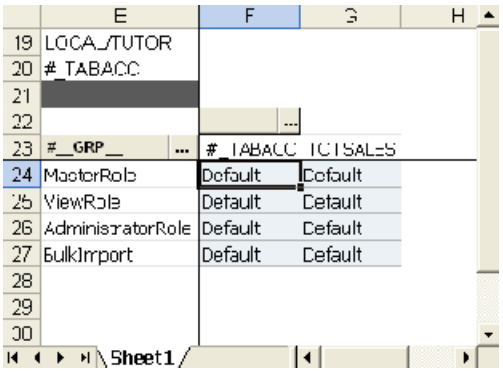
Before a user is assigned a specific data access right to a OLAP Server cube or area within a OLAP Server cube, that user has DEFAULT rights. The OLAP Server administrator can determine what the DEFAULT rights are for a specific cube (READ, WRITE or NONE).

Cube access control cube

Caution: If a user has the permission Administer OLAP Server Database assigned in the OLAP Permission Management, no data access rights are checked.

Whenever you create an OLAP database, an OLAP cube called #_TABACC will be generated for the database. This is a two-dimensional cube containing a list of all the roles as one dimension and a list of all available cubes as the second dimension. In it you can assign READ, WRITE, NONE or DEFAULT

rights for OLAP Server roles and a cube. Simply enter the appropriate access level in the cell located at the intersection of the cube name and role you want to administer.

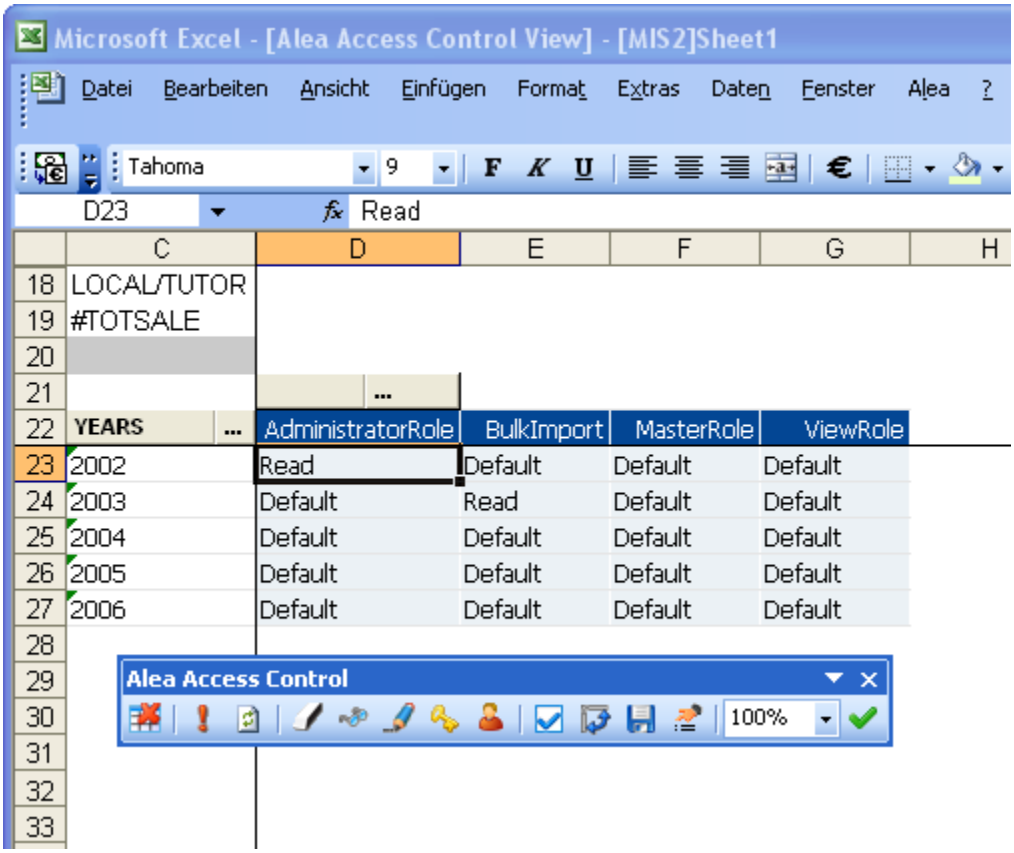


Only users with administrator rights have access to this cube. You must use Repository Administration to assign general access rights like administrator rights.

Dimension access control (DAC) cubes

> **Caution:** If a user has the permission Administer OLAP Server Database assigned in the OLAP Permission Management, no data access rights are checked.

For each dimension in a OLAP Server database, you can create a Dimension Access Control (DAC) cube. This is a two dimensional cube containing a list of all the user groups as one dimension and the dimension whose access you want to restrict as the second dimension. In it you can assign READ, WRITE, NONE or DEFAULT rights for an OLAP Server role and element. Enter the appropriate access level in the cell located at the intersection of element name and user group you want to administer.



Only users with administrator rights have access to this cube.

You must use Repository Administration to assign general access rights like administrator rights.

The permissions are represented in the DAC cube cells by these numeric values:

Permission	Cell value
DEFAULT	Empty cell
NONE	0
READ	1
WRITE	2
NONE, to be passed	8
READ, to be passed	9
WRITE, to be passed	10
NONE, inherited	16
READ, inherited	17
WRITE, inherited	18

Permissions that should be passed are higher by 8 then the permission itself. For example, `READ, to be passed` is 9 which is 1 + 8.

Inherited permissions are higher by 16 then the permission itself. For example, `WRITE, inherited` is 18 which is 2 + 16. The inherited permissions are created by the OLAP Server. They cannot be written.

Multidimensional access control

Caution: If a user has the permission Administer OLAP Server Database assigned in the OLAP Permission Management, no data access rights are checked.

You can create a cube for multidimensional access control (Multi-DAC) for several dimensions in a OLAP Server database. This is a multidimensional cube containing a dimension of all the OLAP Server roles as one dimension and all the other dimensions for which the access is to be restricted. In this cube you can assign READ or WRITE rights for an OLAP Server role and for each dimension element.

You can assign a cube not only one, but several Multi-DACs, but you need to have WRITE access to all those Multi-DACs. For details about assigning Multi-DACs refer to Assigning Data Access Rights to Multiple Dimensions.

Only users with administrator rights have access to this cube. You must use Repository Administration to assign general access rights like administrator rights.

The permissions are represented in the MDAC cube cells by these numeric values:

Permission	Cell value
DEFAULT	Empty cell
NONE	0
READ	1
WRITE	2

Assigning data access rights

Assigning data access rights to cubes

Before assigning data access rights to cubes, you must create the appropriate OLAP Server roles in Repository Administration. To assign access rights to cubes in a database, you must have Administrator rights to that database. To assign access rights to a cube, you must first activate it for access control, then you can control the access to cubes via the TABACC cube.

Step 1: Activating the access control

1 Select **Infor BI Office Plus > Database > Cubes and Data Areas**. Or, click the **Cubes** button on the File and Database toolbar. The Cubes and Data Areas dialog is displayed.
2 Select the cube for which you want to activate access control.
3 Click the **Properties** button.
4 The Cube Properties dialog is displayed. Go to the **Access Permissions** tab.
5 Select the **Enable access control via the #_TABACC cube** check box.

Step 2: Assigning access rights via the #_TABACC cube

1 Open the #_TABACC cube by double-clicking it in the Cubes and Data Areas dialog. Within this cube you can assign access rights for all cubes that have been activated for access control. You can assign Data Access Rights (Read, Write, None, Administrator, or Default) to all the OLAP Server roles.

   See "Data access rights" on page 161.

2 Click the cell at the intersection of the cube and the role in question.

   If you have created new roles in Repository Administration, you must first update your #_GRP_ dimension, before you can assign access rights to those new roles. Click the **Update Roles** button on the Alea Access Control toolbar to accomplish this.

   After updating the #_GRP_ dimension, you must close the #_TABACC cube and reopen it.

3 On the Infor BI Alea Ad-hoc Report toolbar, choose the access level you want to assign from the **Permissions** list.

Toolbar button	Description
	Assigns no access rights to the selected element.
	Assigns Read rights to the selected element.
	Assigns Write rights to the selected element.
	Assigns Administrator rights to the selected element.
	Assigns Default rights to the selected element.

Assigning data access rights to single dimensions

Before assigning data access rights, you must create the appropriate OLAP Server roles in Repository Administration. To assign access rights to dimension elements, you must have Administrator rights to the database containing the dimension. To assign access rights to a dimension, you must first activate it for access control by creating a DAC cube for it. You may then assign access rights to the dimension elements via this cube.

Step 1: Creating a DAC cube for a dimension

1 Open the Dimensions and Subsets dialog by selecting **Infor BI Office Plus > Database > Dimensions and Subsets**. Or, by clicking the **Dimensions** button on the Infor BI Alea Ad-hoc Report toolbar.
2 Select the dimension for which you want to create a DAC.
3 Click the **Properties** button. The Dimension Properties dialog is displayed.
4 Go to the **Access Permissions** tab.
5 Click the **New Cube** button. In the New DAC Cube dialog you can either accept the suggested name or type another name. We recommend that you name all DAC cubes using the letters 'DAC' and the first five letters of the dimension that the DAC cube is controlling. A DAC cube name may have a maximum of eight characters.
6 Click **OK**. Your DAC cube is displayed in the list.

Step 2: Assigning access rights via the DAC cube

1 Select **Infor BI Office Plus > Database > Cubes and Data Areas**. Or, click the **Cubes** button on the File and Database toolbar. The Cubes and Data Areas dialog is displayed.
2 Double-click the DAC cube you want to edit. In this cube you can assign access rights for all dimension elements.
3 Select the cell at the intersection of the element and the OLAP Server role in question.

   Note: If you have created new roles in Repository Administration, you must first update your #_GRP_ dimension, before you can assign access rights to those new roles. Click the **Update Roles** button on the Alea Access Control toolbar to accomplish this. After updating the #_GRP_ dimension you must close the DAC and reopen it.

4 On the Infor BI Alea Ad-hoc Report toolbar, choose the access level you want to assign from the **Permissions** list.

Toolbar button	Description
	Assigns no access rights to the selected element.
	Assigns Read rights to the selected element.
	Assigns Write rights to the selected element.
	Not supported for DAC.
	Assigns Default rights to the selected element.

Assigning data access rights to multiple dimensions

Before assigning data access rights, you must create the appropriate OLAP Server roles in Repository Administration. To assign multidimensional access rights in a database, you must have Administrator rights to that database. In order to assign access rights to a Multi-DAC, you must first create one.

Step 1: Creating or assigning a Multi-DAC

1 Select **Infor BI Office Plus > Database > Cubes and Data Areas**. Or, click the **Cubes** button on the File and Database toolbar. The Cubes and Data Areas dialog is displayed.
2 Select the cube for which you want to create a Multi-DAC.
3 Click the **Properties** button.
4 The Cube Properties dialog is displayed. Go to the **Access Permissions** tab.
5 If you want to assign the selected cube a Multi-DAC, select it from the list. If you want to create a Multi-DAC, click the **New Cube** button.
6 In the **New Cube for Multidimensional Access Permissions on 'Name'**, select the desired dimensions.

Step 2: Assigning access rights via the Multi-DAC

1 Open your Multi-DAC in the Cubes and Data Worlds dialog.
2 In the #_GRP_ dimension you can select the OLAP Server roles.

   If you have created new roles in Repository Administration, you must first update your #_GRP_ dimension, before you can assign access rights to those new roles. Click the **Update Roles** button on the Alea Access Control toolbar to accomplish this. After updating the #_GRP_ dimension, you must close the Multi-DAC and reopen it.

3 Arrange the dimensions and elements on your spreadsheet to assign the desired rights to all the elements.

Toolbar button	Description
	Assigns no access rights to the selected element.
	Assigns Read rights to the selected element.
	Assigns Write rights to the selected element.
	Not supported for Multi-DAC.
	Assigns Default rights to the selected element.

Differences between DAC and Multi-DAC

Rights defined in a DAC are applied to dimension elements.

Rights defined in a Multi-DAC are applied on cells or subcubes.

If there are more access tables, they are combined and the most restrictive right is effective for each role. If a user has more than one role, they have the rights of the least restricted role.

Standard log on to unprotected OLAP Server databases

When a OLAP Server database is created, it is automatically assigned a standard user ID and password. These are:

- User ID: ADMIN
- Password:

This means that a default user is always created with Administrator rights, with the user name ADMIN and no password. They are set as the default log on user ID and password in every version of OLAP Server. This is important when databases are distributed to others where access control is an issue. Be sure to either change the password of this default user before distributing the database, or replace this user with another user name with administrator privileges.

User management integration using Repository

Users for all Infor BI products are administered in the Repository. The Repository is a relational database. Microsoft and Oracle are supported.

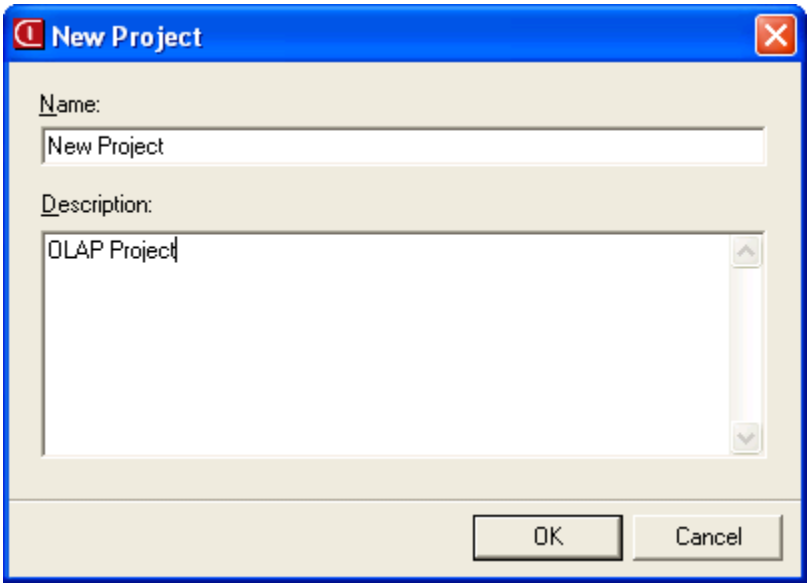
New OLAP Server users and roles are created and maintained in the OLAP Permission Management node of Repository Administration.

Permission management

Start Repository Administration by selecting **Start > All Programs > Infor BI > Infor BI Repository Administration**.

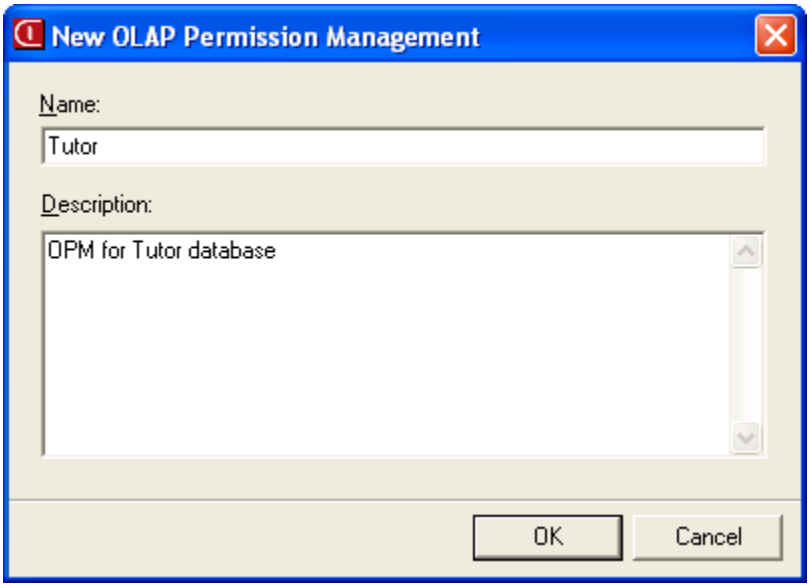
Creating a project

Right-click Projects and choose **New Project**. Enter a name and a description.



Creating a permission management

Right-click OLAP Permission Management and choose **New OLAP Permission Management**. Enter a name and a description.



Users

With a default installation you will find four initial users in the User Management: Admin, Guest, AleaInternal, and SchedulerInternal.

Admin

The Admin is entitled to administer users and groups within the User Management. The Admin has got the Administrator and Edit Password role and is assigned to the Every SSO User group. This user cannot be deleted.

Guest

The Guest may only modify their password, therefore they are assigned the Edit Password role. This user belongs to the Every SSO User group.

AleaInternal

The AleaInternal represents the account for the Alea Server and is used for the communication to the Common Object Store. This account is a static object that cannot be deleted. The AleaInternal can be assigned any role and belongs to the Every SSO User group.

SchedulerInternal

The SchedulerInternal represents the account for the Task Manager in onVision and is assigned the Log On As role. This account is a static object that can neither be changed nor deleted. This user belongs to the Every SSO User group.

Roles

This tab provides the possibility to create new roles and to assign them permissions as well as users and groups. There are four initial roles for OLAP Server projects:

- AdministratorRole

  This role holds the Administer, the Administer OLAP Server Database and the View permission and is assigned to the Admin and the AleaInternal.

- BulkImport

  This role entitles the holder to suppress the generation of events in the Event Agent.

- DesignerRole

  This role contains the OLAP permissions Delete, Edit, View, and Write Values. The role is assigned to the Report Designer group.

- MasterRole

  This role includes all OLAP Server permissions and is assigned to the Admin user.

- ViewRole

  The ViewRole holds the View permission and is assigned to the Admin, the AleaInternal and to the Every SSO User group.

Assigning Permissions to OLAP Server Roles

Expand the view for the role and click OLAP Server Permissions. Click the Assign OLAP Server Permissions shortcut menu command. Make your choice from the OLAP Server Permissions list. Click the **Select all** button for a quick selection or the **Deselect all** button or the opposite action.

Assigning User Management Objects to Roles

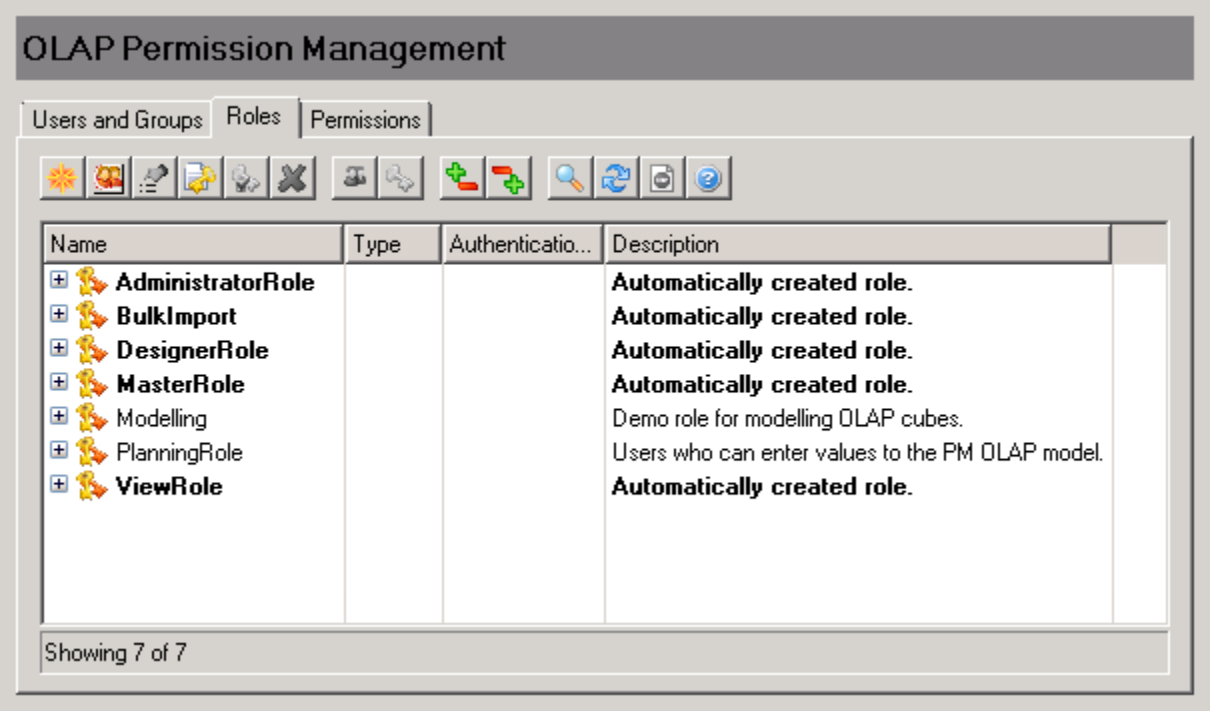
Expand the view for the role in question and then expand User Management Objects. Select the **Assign User Management Objects** shortcut menu command. Make your choice in the User Management objects list. Click the **Select all** button for a quick selection and the **Deselect all** button or the opposite action.

Editing the Role Properties

Click the role and click the **Properties** command. Change the description and confirm by **OK**.

Deleting Roles

Right-click the role in question and choose the **Delete** command.



Permissions

It is not possible to create new permissions or to change the existing OLAP Server permissions or their descriptions. You can assign permissions to or remove them from roles and you can also change the descriptions of roles.

In the OLAP Permission Management there are the following OLAP Server permissions:

OPM administration permissions

OLAP server permission	Description
Administer	Allows the bearer to administer OLAP Server permissions. This OLAP Server permission is assigned to the OLAP Server roles AdministratorRole and MasterRole.

OLAP server permission	Description
Delete	Allows the bearer to delete the OLAP Permission Management. Both the OLAP Permission Management or its subordinate elements must not be checked out by other users. This OLAP Server permission is assigned to the OLAP Server role MasterRole.
Edit	Allows the bearer to edit the properties of the OLAP Permission Management. The project must not be checked out by another user. This OLAP Server permission is assigned to the OLAP Server role MasterRole.
View	Allows the bearer to open and view the OLAP Permission Management. This OLAP Server permission is assigned to the OLAP Server roles AdministratorRole, MasterRole, and ViewRole.

OLAP Server database permissions

OLAP server permission	Description
Administer OLAP Server Database	Allows the bearer to administer the OLAP Server database. This OLAP Server permission is assigned to the OLAP Server roles AdministratorRole and MasterRole.
Edit Dimensions	Allows the bearer to edit dimensions in OLAP Server. This OLAP Server permission is assigned to the OLAP Server role MasterRole.
Edit Rules	Allows the bearer to edit rules in OLAP Server. This OLAP Server permission is assigned to the OLAP Server role MasterRole.
Import/Export Values	Allows the bearer to import and export values in OLAP Server. This OLAP Server permission is assigned to the OLAP Server role MasterRole.
Start Database	Users with this permission, or with Administer ComManager permission on a project, can start of stop OLAP databases.
Write values	Allows the bearer to write values in OLAP Server. This OLAP Server permission is assigned to the OLAP Server role MasterRole.

Caution: If a user has the permission Administer OLAP Server Database assigned in the OLAP Permission Management, no data access rights are checked.

See "Data access rights" on page 161.

Assigning OLAP Server Permissions to Roles

Expand the view for the permission in question and click **OLAP Server Roles**. Select the **Assign OLAP Server Roles** command. Make your choice in the OLAP Server roles list. Click the **Select all** button for a quick selection and the **Deselect all** button or the opposite action and confirm by **OK**.

Removing Roles

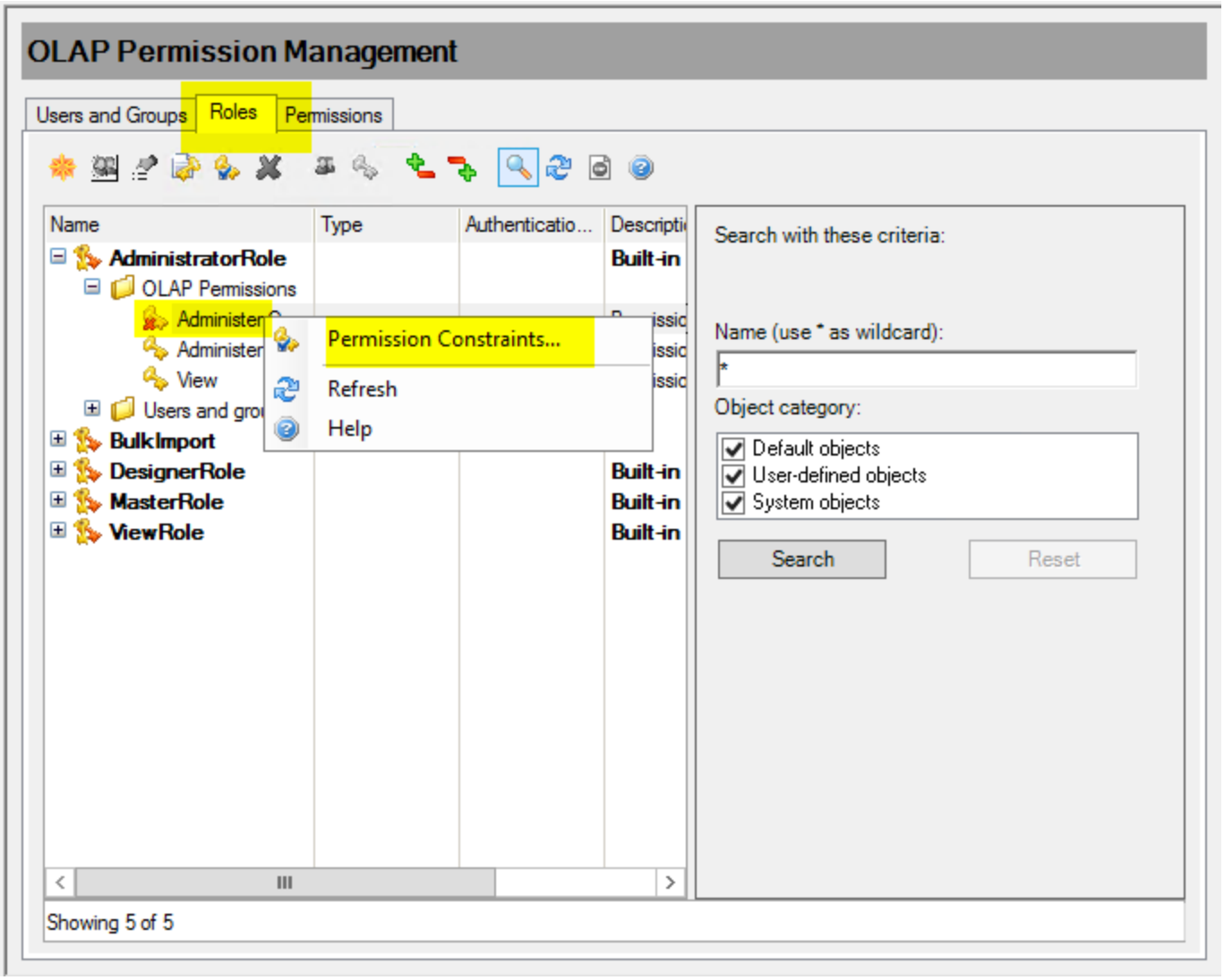
Right-click the role in question and select the **Remove OLAP Server Role** command or on the toolbar.

Restricting the permissions of a role to certain applications (constraints)

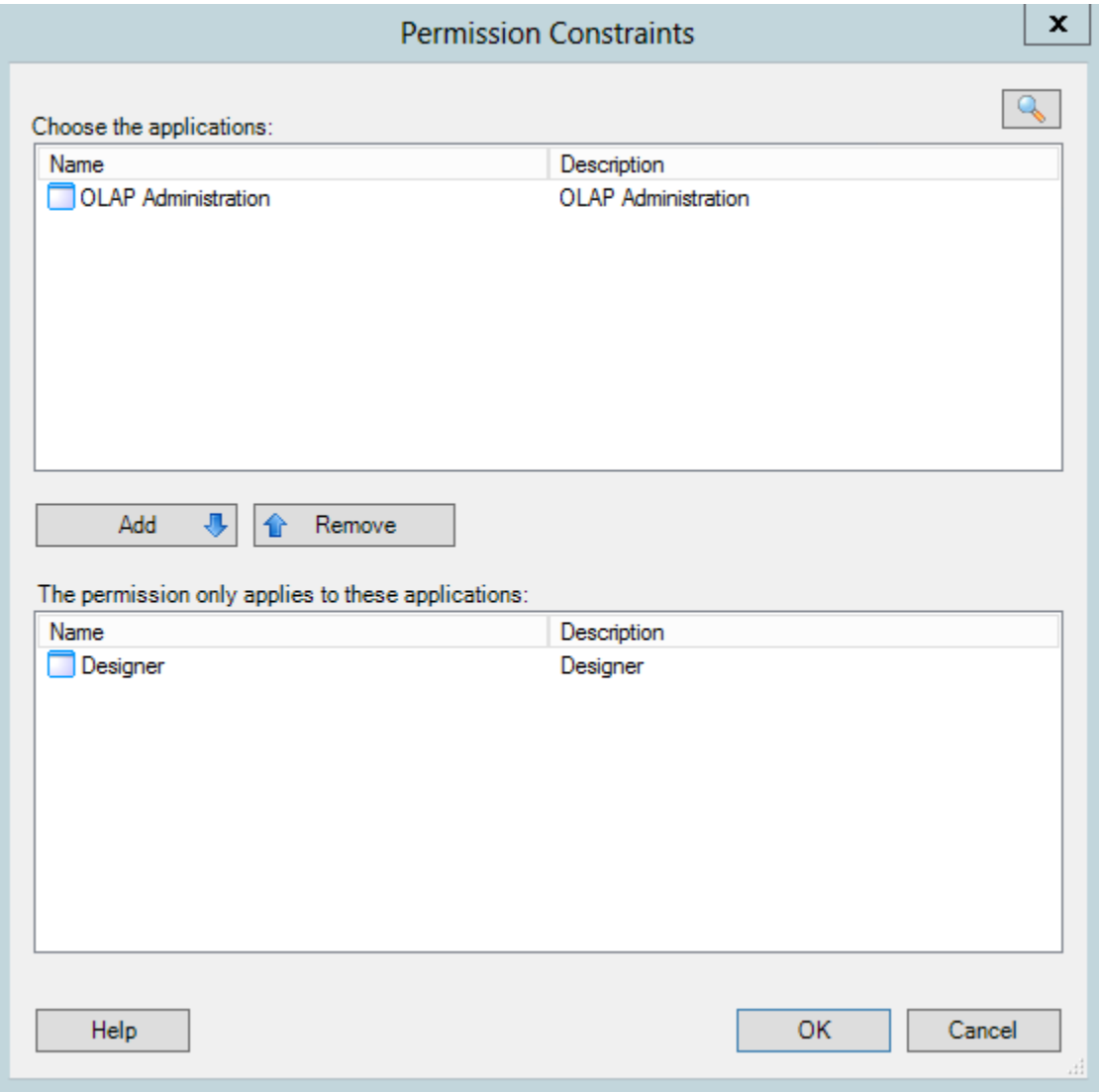
For details on this topic refer to the Repository Administration online help.

The permissions of a role can be different, depending on the application a user logs on to (constraints of permissions). Users can inherit different permissions by the same role for different applications. To restrict one of the permissions of a role to a specific application, you assign the application, for which the permission is valid, in the Permission Constraints dialog.

Note: The applications for restrictions are fixed in the database. By default, if you do not specify a constraint, the permission is valid for all applications.



To open the Permission Constraints dialog, expand the OLAP Permissions folder under a role. Right-click the permission and select the **Permission Constraints** command on the shortcut menu or on the toolbar. You can only open the dialog, if applications are configured for constraints in the database.



In the upper part, you see the applications that are not assigned to the permission. The list below contains the assigned applications. Double-click the applications to move them from one list to the other. To assign or to remove multiple applications, select them in the lists (multiple selection with Ctrl) and click **Add** or **Remove**.

LoadFromSource

13

This chapter describes the structure of the relational database from which the OLAP database can be loaded.

The metadata to transfer are stored in tables with the following names:

The data are stored in tables which have names set by the user. These tables are categorized as:

- Member tables (or element tables)
- Member relationship tables (or element relationship tables or element hierarchy tables)
- Subset tables
- Fact tables (or data tables)

Setup for process

When the OLAP Server starts up it reads the `Db.ini` file. In this file are contained control data for the LoadFromSource process. There are two parameters for controlling the loading process.

- In `[INIT]` section: `RelationalConnectionString`

  This is the connection string to the relational database. Its form will be in the syntax required by this database. It needs to be present only for loading process.

  The connection string can be defined in the OLAP Administration by selecting **Database Settings > LoadFromSource Database > Connection String**.

  Example of connection strings:

  - SQL Server

    ```
    Driver={SQL Server Native Client
    10.0};Server=localhost\sql2008;Database=MY_DB;Trusted_Connection=yes;Pooling=false
    ```

  - Oracle

    ```
    DRIVER={Oracle in OraDb11g_DB}; DSN=Oracle Datasource; DBQ=ORCL;
    UID=SYSTEM; PWD=Password
    ```

  - PostgreSQL

    ```
    DRIVER={PostgreSQL ODBC Driver(Unicode)}; SERVER=czpr-sqlserver2;
    PORT=5433; DATABASE=Demo_DB; UID=system; PWD=Password
    ```

  As OLAP Server just passes the connection string to the ODBC library, you need to provide whatever is needed by the ODBC driver.

- In `[SCHEDULER]` section: `DBLoad`
  - Need not be present. The default value is NO.
  - NO means do not trigger the loading process automatically at start up. This is the default value if the `DBLoad` parameter is not present.
  - YES means trigger the loading process at start up. Furthermore the _Jobs table is checked in a configurable interval (see "Db.ini" on page 271) for new data to be loaded. Any new jobs represented by rows in the _Jobs table for which Status = NULL will be executed.

Triggering the process

Console window

Enter the command:

```
dbload
```

OLAP Administration

Connect to a database and select **Trigger LoadFromSource Database** in the context menu of the database.

OLAP API using an XML request

IN:

```
<Alea:Document>
   <Alea:Request RequestID="001" Class="Database" Method="StartDBLoad"
 />
</Alea:Document>
```

OUT:

```
<Alea:Document>
   <Alea:Request RequestID="001">
      <Alea:Return/>
   </Alea:Request>
</Alea:Document>
```

on Error:

```
<Alea:Document>
   <Alea:Request RequestID="001">
      <Alea:Error ErrorID="error_code"/>
   </Alea:Request>
</Alea:Document>
```

VB-API

```
Dim mdsRet As Variant
mdsRet = StartDBLoad("Local/Tutor")
```

Notes

- The SQL syntax used is for both SQL Server and Oracle. Amazon Redshift (Postgres) is also handled. This syntax for this is the same as for SQL Server.
- A comment column is present in every metadata table for the user's convenience. It is not processed by the server. It is for the user to add notes relating to the rows they refer to. Although this column appears in each CREATE TABLE command, it is not referred to in the table descriptions.
- Element names can only have string values. This will change in later versions.
- Table names defined by the user cannot contain spaces or be reserved words. They can be sub-queries.
- Table names that are not sub-queries can be escaped (but need not be).
- Sub-queries used in place of tables must not be escaped.
- Do not use whitespace characters in sub-queries other than the standard space character " ". (Do not use tabs or carriage returns, for example.)
- Do not use an alias with a table name.
- The portion of a sub-query preceding any alias must be placed between parentheses "(" and ")". This will mean the whole query if no alias is present.

Metadata Tables

Because of potential issues with joins in views, we recommend that you use tables and not views for relational loads from Postgres sources. In Postgres, column names are case-sensitive. We recommend that you always use quotes in requests for table and column names. Unquoted table and column names are rendered in lowercase.

_Jobs

_Jobs stores job data. One row describes one job.

Column	Type	Description
JobId	Big Integer	Sequential key identifying job and ordering jobs
JobType	Integer	Enumerator for kind of job. 1 = Load Dimension

Column	Type	Description
		2 = Create Cube 3 = Load Cube 4 = Set Cube Properties 5 = Set Dimension Properties 6 = Fill Dimension Attributes 7 = Set Cube Rules 8 = Load Subsets 9 = Clear Cube Region 10 = Delete Dimension 11 = Delete Cube 12 = Set Dimension Level Names 13 = Load Dimension Create DAC 14 = Load Elements 15 = Set Attribute Driven Hierarchies
JobObject	String(71)	The dimension or cube that the job will operate on for job types 1, 2, 4, 5, 6, 7, 8, 10 , 11, 12, 13, 14, 15. For job types 3 and 9 this is set to JobObject of FactLoadParameters.
JobParameter	Integer	Key to _JobsParameters table, which gives element filter information for job.
JobGroup	String(100)	Reports that display error logs use the JobGroups column to combine multiple jobs. OLAP Server does not process this information.
Started	DateTime	Start date-time of process
Completed	DateTime	Completion date-time of process
Status	Integer	Completion status of process. This is the integer error status from the kernel. **Note:** 0 = Success NULL = Unexecuted See "Related error messages" on page 228.

Comments

- The OLAP Server reads through the un-executed jobs. These are indicated by a Status value of NULL.
- The jobs are executed in JobId order.
- At start, the value of the Started column is set to the date-time of the start of the job.
- After completion, the value in the Completed column is set to the date-time of completion and the Status is set to the error condition from the kernel of the server. The value for success is 0.
- The columns Started and Completed date/times are given in:
  - Normal time if LogTimeInUTC in `Db.ini` is set to NO
  - UTC time if LogTimeInUTC in `Db.ini` is set to YES
- Rows from this table are not deleted by the server.
- To delete the cube rules, send an empty rules tag. For example:

```
<Alea:Rules xmlns:Alea="http://www.misag.com" />
```

Note: If one job fails during the load of the meta data or the execution of the job:

- The job receives an appropriate error code describing the failure (for example connection cannot be established, meta data is incorrect, element does not exist).
- All other jobs are executed, the failure of one job has no effect on other jobs.

_Jobs Table Creation with SQL Server

```
CREATE TABLE [dbo].[_Jobs](
   [JobId] [bigint] NOT NULL UNIQUE,
   [JobType] [int] NOT NULL,
   [JobObject] [nvarchar](71) NOT NULL,
   [JobParameter] [int] NULL,
   [JobGroup] [nvarchar](100) NULL,
   [Started] [datetime] NULL,
   [Completed] [datetime] NULL,
   [Status] [int] NULL,
   [Comment] [ntext] NULL
)
```

_Jobs Table Creation with Oracle

```
CREATE TABLE "_Jobs"(
   "JobId" number(19, 0) NOT NULL UNIQUE,
   "JobType" int NOT NULL,
   "JobObject" nvarchar2(71) NOT NULL,
   "JobParameter" int NULL,
   "JobGroup" nvarchar2(100) NULL,
   "Started" timestamp(6) NULL,
   "Completed" timestamp(6) NULL,
```

```
   "Status" int NULL,
   "Comment" nclob NULL
);
```

_Jobs Table Creation with Postgres

```
CREATE TABLE "_Jobs"(
   "JobId" bigint NOT NULL UNIQUE,
   "JobType" int NOT NULL,
   "JobObject" varchar(71) NOT NULL,
   "JobParameter" int NULL,
   "JobGroup" varchar(100) NULL,
   "Started" timestamp NULL,
   "Completed" timestamp NULL,
   "Status" int NULL,
   "Comment" text NULL
);
```

_ScheduledJobs

Stores the details of jobs that are to be scheduled.

Column	Type	Description
Id	Big Integer	Sequential key identifying scheduled job. **Note:** This is not the same as JobId of _Jobs table.
JobType	Integer	Enumerator for kind of job. 1 = Load Dimension 2 = Create Cube 3 = Load Cube 4 = Set Cube Properties 5 = Set Dimension Properties 6 = Fill Dimension Attributes 7 = Set Cube Rules 8 = Load Subsets 9 = Clear Cube Region 10 = Delete Dimension 11 = Delete Cube 12 = Set Dimension Level Names

Column	Type	Description
		13 = Load Dimension Create DAC
JobObject	String(71)	The dimension or cube that the job will operate on for job types 1, 2, 4, 5, 6, 7, 8, 10, 11, 12 and 13. For job types 3 and 9 this is set to JobObject of FactLoadParameters.
JobParameter	Integer	Key to _JobsParameters table, which gives element filter information for job.
Schedule	String(200)	The schedule to which a job is to be created. This is expressed in `cron` format. Currently supported cron strings: *: wildcard for every minute, hour, ... */n: interval of n minutes, hours, ... m: single point in time. For example, minute m, hour m, ... m,n,o: list of single points in time
Scheduled	Boolean	True/False flag to indicate whether a job will be scheduled, that is, whether a new row will be added to the _Jobs table.
Disabled	Boolean	True/False flag to indicate whether this scheduled job will or will not trigger the adding of a job in the _Jobs table. If this is True, no new job will be created based on this scheduled job.

Comments:

- The OLAP Server reads through this table to decide if any new jobs will be added to the _Jobs table.
- This table is only used if `DBLoad` is set to YES in `Db.ini`.
- It is read through repeatedly after an interval of up to approximately a minute.
- A new job in _Jobs will be created if the field Scheduled is set to True. This will happen if all the following criteria are met:
  - The field Disabled is False. (If this is True, the load process looks no further: a new job based on the current row of _ScheduledJobs will not be created if Disabled = True.)

  - If Disabled is False, then the Schedule field, a `cron` string, is checked for a match with the current time. If there is a match, the Boolean Scheduled field is then set to True. (It is set to False otherwise.)
  - If Scheduled is already True before this process had begun, it is left with this value. (Scheduled can only be True if a job was scheduled to run on an earlier occasion but had not yet had the chance to.)

- The new job created will take the next available JobId in the _Jobs table, that is, the current maximum plus 1.
- The values JobType, JobObject and JobParameter will be copied into the fields of the same name of the _Jobs table.
- After execution of a spawned job, a note is appended to the field Comment of _Jobs to indicate what had been loaded.
- After all spawned jobs have executed, the field Scheduled is set to False for all the rows of _ScheduledJobs from which the new rows in _Jobs were created in the current job run.
- When the LoadFromSource scheduler is enabled, there is an additional polling of the _ScheduledJobs table every 43 seconds

  If the time matches for Scheduled Job to be created, a row is written to the _Jobs table to be executed.

OLAP Server behavior in case of problems loading meta data of a job

In the case that error occurs during the load of the meta data or the execution of the job:

- The failed job receives an appropriate error code describing the failure, for example, `Connection cannot be established`, `Incorrect meta data`, `Element does not exist`.
- If a job fails, this has no direct effect on other jobs.

_ScheduledJobs Table Creation with SQL Server

```
CREATE TABLE [dbo].[_ScheduledJobs](
   [Id] [bigint] NOT NULL UNIQUE,
   [JobType] [int] NOT NULL,
   [JobObject] [nvarchar](71) NOT NULL,
   [JobParameter] [int] NULL,
   [Schedule] [ntext] NOT NULL,
   [Scheduled] [bit] NOT NULL,
   [Disabled] [bit] NOT NULL,
   [Comment] [ntext] NULL
)
```

_ScheduledJobs Table Creation with Oracle

```
CREATE TABLE "_ScheduledJobs"(
   "Id" number(19, 0) NOT NULL UNIQUE,
   "JobType" int NOT NULL,
   "JobObject" nvarchar2(71) NOT NULL,
   "JobParameter" int NULL,
   "Schedule" nclob NOT NULL,
   "Scheduled" number NOT NULL,
```

```
   "Disabled" number NOT NULL,
   "Comment" nclob NULL
);
```

_ScheduledJobs Table Creation with Postgres

```
CREATE TABLE "_ScheduledJobs"(
   "Id" bigint NOT NULL UNIQUE,
   "JobType" int NOT NULL,
   "JobObject" varchar(71) NOT NULL,
   "JobParameter" int NULL,
   "Schedule" text NOT NULL,
   "Scheduled" int NOT NULL,
   "Disabled" int NOT NULL,
   "Comment" text NULL
);
```

_Sources

Describes the data sources of OLAP objects (dimensions and cubes) loaded with the DbLoad mechanism.

The other metadata tables contain a column called SourceID. The ID read from the other tables is used to find the correct row in the _Sources table. If there is no entry, the OLAP Server finds the appropriate table or view in the same database as the metadata tables. When looking up the source in the _Sources table, the OLAP Server reads the source type and the connection parameters. Depending on the type, this can be either a direct connection string or the ID used to retrieve a connection string from a global data source

Column	Type	Description
SourceId	Integer	Identifier for the source row in this table.
SourceType	Integer	Enumerator for type of source: 1 = SQL Server, Oracle or Amazon Redshift (Postgres) ODBC 2 = Aqua ODBC (Not yet implemented.) 3= The connection string is obtained from Global Data Sources in the Repository
ConnectionParameters	Text	Connection string to the source or a data source identifier, including tenant prefix, used to look up the required connection string from Global Data Sources in the Repository .

_Sources Table Creation with SQL Server

```
CREATE TABLE [dbo].[_Sources](
   [SourceId] [int] NOT NULL,
   [SourceType] [int] NOT NULL,
   [ConnectionParameters] [ntext] NULL,
   [Comment] [ntext] NULL
)
```

_Sources Table Creation with Oracle

```
CREATE TABLE "_Sources"(
   "SourceId" int NOT NULL,
   "SourceType" int NOT NULL,
   "ConnectionParameters" nclob NULL,
   "Comment" nclob NULL
);
```

_Sources Table Creation with Postgres

```
CREATE TABLE "_Sources"(
   "SourceId" int NOT NULL,
   "SourceType" int NOT NULL,
   "ConnectionParameters" text NULL,
   "Comment" text NULL
);
```

_Dimensions

_Dimensions describes metadata relating to the dimensions. One row contains data at dimension level for one dimension.

Column	Type	Description
DimensionName	String(50)	Name of the dimension
DimensionDescription	String(150)	Description of the dimension
SourceId	Integer	Data source of the dimension. This links to the key SourceId of the table _Sources which describes the details of the source. NULL means the dimension source uses the same connection as in `Db.ini`.

Column	Type	Description
Param1	Text	First parameter. For DimensionSource = 1, this is the relational table source of the dimension data, the members' table. This can be a sub-query.
Param2	Text	Second parameter. For DimensionSource = 1, this is the relational table source of the dimension relations data. This might be the same table as the members' table found in Param1. NULL if there are no relations for the relational source. This can be a sub-query.
DimensionType	Integer	Enumerator for the ODBC type of the dimension. 0 = Unknown 1 = Time 2 = Measure 3 = Other (4 is not used) 5 = Quantitative 6 = Accounts 7 = Customers 8 = Products 9 = Scenario 10 = Utility 11 = Currency 12 = Rates 13 = Channel 14 = Promotion 15 = Organization 16 = Bill Of Materials 17 = Geography
DefaultElement	String(71)	If an element of a dimension is not specified, take it to be this one. NULL if there is no default element. If specified default element is not present in dimension, no default element property is

Column	Type	Description
		created and a warning is issued in the log file.
FlatView	Boolean	Flags if the view of the hierarchy of the members of the dimension is flat or not. True = Flat; use straight list of elements that does not show hierarchy, although parent elements can still be expanded to show their children. False = Not flat; use expanded hierarchical diagram.
InvertedHierarchy	Boolean	Flags if the hierarchy of the member is displayed inverted. True = Show Inverted Hierarchy, that is, show parents below list of children. False = Show Hierarchy with parents first and children listed below them.
AccessCube	String(50)	Name of the access cube of the dimension. NULL if the dimension has no access cube.
LoadFlags	BigInteger	Flags controlling the dimension load. The following powers of 2 flag the action indicated: 1 = Do not break the import on the first unresolved error that occurs. 2 = Document each error that occurs. • · Combinations of these can be made adding their values together. • Default value is 0. A NULL value is interpreted as 0.
ExtendedProps	Text	The extended properties of the dimension expressed as an XML document string. NULL means the dimension has no extended properties.

Comments

- Rows of the _Dimensions table are read into memory. The rows are those for which there is a loading operation to carry out in the _Jobs table.
- These _Dimensions rows are linked from the _Jobs table through the column JobObject via DimensionName of _Dimensions.

_Dimensions Table Creation with SQL Server

```
CREATE TABLE [dbo].[_Dimensions](
   [DimensionName] [nvarchar](50) NOT NULL,
   [DimensionDescription] [nvarchar](150) NOT NULL,
   [SourceId] [int] NULL,
   [Param1] [ntext] NULL,
   [Param2] [ntext]NULL,
   [DimensionType] [int] NULL,
   [DefaultElement] [nvarchar](71) NULL,
   [FlatView] [bit] NOT NULL,
   [InvertedHierarchy] [bit] NOT NULL,
   [AccessCube] [nvarchar](50) NULL,
   [LoadFlags] [bigint] NULL,
   [ExtendedProps] [ntext] NULL,
   [Comment] [ntext] NULL
)
```

_Dimensions Table Creation with Oracle

```
CREATE TABLE "_Dimensions"(
   "DimensionName" nvarchar2(50) NOT NULL,
   "DimensionDescription" nvarchar2(150) NOT NULL,
   "SourceId" int NULL,
   "Param1" nclob NULL,
   "Param2" nclob NULL,
   "DimensionType" int NULL,
   "DefaultElement" nvarchar2(71) NULL,
   "FlatView" number NOT NULL,
   "InvertedHierarchy" number NOT NULL,
   "AccessCube" nvarchar2(50) NULL,
   "LoadFlags" number(19, 0) NULL,
   "ExtendedProps" nclob NULL,
   "Comment" nclob NULL
);
```

_Dimensions Table Creation with Postgres

```
CREATE TABLE "_Dimensions"(
   "DimensionName" varchar(50) NOT NULL,
   "DimensionDescription" varchar(150) NOT NULL,
   "SourceId" int NULL,
   "Param1" text NULL,
```

```
   "Param2" text NULL,
   "DimensionType" int NULL,
   "DefaultElement" varchar(71) NULL,
   "FlatView" int NOT NULL,
   "InvertedHierarchy" int NOT NULL,
   "AccessCube" varchar(50) NULL,
   "LoadFlags" bigint NULL,
   "ExtendedProps" text NULL,
   "Comment" text NULL
);
```

_DimensionLevelNames

_DimensionLevelNames sets the level names of a dimension.

If this table is absent, no level names are set. If the table is present, the name of each defined level is set in the dimension. Levels not listed are not touched.

Column	Type	Description
DimensionName	String(50)	Refers to the _Dimensions table.
LevelNumber	Integer	The level to address. These numbers are zero-based and counted top down by OLAP.
LevelName	String(50)	The new name to assign to the level. NULL or empty removes an already assigned level name.

_DimensionLevelNames Creation with SQL Server

```
CREATE TABLE [dbo].[_DimensionLevelNames](
   [DimensionName] [nvarchar](50) NOT NULL,
   [LevelNumber] [int] NOT NULL,
   [LevelName] [nvarchar](50) NULL,
   [Comment] [ntext] NULL
)
```

_DimensionLevelNames Creation with Oracle

```
CREATE TABLE "_DimensionLevelNames"(
   "DimensionName" nvarchar(50) NOT NULL,
   "LevelNumber" int NOT NULL,
   "LevelName" nvarchar(50) NULL,
   "Comment" ntext NULL
);
```

_DimensionLevelNames Creation with Postgres

```
CREATE TABLE "_DimensionLevelNames"(
   "DimensionName" nvarchar(50) NOT NULL,
   "LevelNumber" int NOT NULL,
   "LevelName" nvarchar(50) NULL,
   "Comment" ntext NULL
);
```

_Cubes

_Cubes describes metadata relating to cubes. One row contains data for one cube at cube level.

Column	Type	Description
CubeName	String(50)	Name of cube.
CubeDescription	String(150)	Description of cube.
Param1	String(150)	Reserved
Param2	String(250)	Reserved

AccessControl	Boolean	Flags whether the cube's data access permissions are controlled by #TABACC or not.
		True = Cube's data access is controlled by #TABACC
		False = Cube's data access is not controlled by #TABACC

TransactionLog	Boolean	Flag for whether the cube has an associated transaction log. True = Has a transaction log. False = Has no transaction log.
CubeType	String(1)	The cube type. Enumerator. Values: U = User cube. D = DAC or MDAC. NULL value means U.
ExtendedProps	Text	Extended properties of the cube expressed as an XML document string. NULL means the cube has no extended properties.
CubeRules	Text	The rules of the cube expressed as an XML document string. NULL means the cube has no rules. NULL does not remove any existing rules.

Comments

- Rows of the _Cubes table are read into memory. These are those rows for which there is a cube creation or loading operation to carry out indicated by a row in the _Jobs table.
- These _Cubes rows are linked from the _Jobs table either:
  - For cube creation or property loading through the column JobObject via CubeName of _Cubes
  - For fact loading via the CubeId column of the _FactLoad table.

_Cubes Table Creation with SQL Server

```
CREATE TABLE [dbo].[_Cubes](
   [CubeName] [nvarchar](50) NOT NULL,
   [CubeDescription] [nvarchar](150) NOT NULL,
   [Param1] [nvarchar](150) NULL,
   [Param2] [nvarchar](250) NULL,
   [AccessControl] [bit] NOT NULL,
   [TransactionLog] [bit] NOT NULL,
   [CubeType] [nchar](1) NULL,
   [ExtendedProps] [ntext] NULL,
   [CubeRules] [ntext] NULL,
   [Comment] [ntext] NULL
)
```

_Cubes Table Creation with Oracle

```
CREATE TABLE "_Cubes"(
   "CubeName" nvarchar2(50) NOT NULL,
   "CubeDescription" nvarchar2(150) NOT NULL,
   "Param1" nvarchar2(150) NULL,
   "Param2" nvarchar2(250) NULL,
   "AccessControl" number NOT NULL,
   "TransactionLog" number NOT NULL,
   "CubeType" nchar(1) NULL,
   "ExtendedProps" nclob NULL,
   "CubeRules" nclob NULL,
   "Comment" nclob NULL
);
```

_Cubes Table Creation with Postgres

```
CREATE TABLE "_Cubes"(
   "CubeName" varchar(50) NOT NULL,
   "CubeDescription" varchar(150) NOT NULL,
   "Param1" varchar(150) NULL,
   "Param2" varchar(250) NULL,
   "AccessControl" int NOT NULL,
   "TransactionLog" int NOT NULL,
   "CubeType" char(1) NULL,
   "ExtendedProps" text NULL,
   "CubeRules" text NULL,
```

```
    "Comment" text NULL
);
```

_CubesDimensions

_CubesDimensions relates each cube to its dimensions.

Column	Type	Description
CubeName	String(50)	Name of cube
DimensionName	String(50)	Name of dimension of the cube whose name is CubeName.
DimensionOrder	Integer	The position of the dimension as it is ordered in the cube.
MeasureDimension	Boolean	Flag for whether the dimension is a measure dimension or not. True = Is a measure dimension. False = Is not a measure dimension. **Caution:** More than one measure dimension on a cube is not permitted. It is the responsibility of the user to ensure that not more than one dimension of a cube is flagged as a measure dimension.

Comment:

For each cube there are as many rows as there are dimensions.

_CubesDimensions Table Creation with SQL Server

```
CREATE TABLE [dbo].[_CubesDimensions](
    [CubeName] [nvarchar](50) NOT NULL,
    [DimensionName] [nvarchar](50) NOT NULL,
    [DimensionOrder] [int] NOT NULL,
    [MeasureDimension] [bit] NOT NULL,
    [Comment] [ntext] NULL
)
```

_CubesDimensions Table Creation with Oracle

```
CREATE TABLE "_CubesDimensions"(
   "CubeName" nvarchar2(50) NOT NULL,
   "DimensionName" nvarchar2(50) NOT NULL,
   "DimensionOrder" int NOT NULL,
   "MeasureDimension" number NOT NULL,
   "Comment" nclob NULL
);
```

_CubesDimensions Table Creation with Postgres

```
CREATE TABLE "_CubesDimensions"(
   "CubeName" varchar(50) NOT NULL,
   "DimensionName" varchar(50) NOT NULL,
   "DimensionOrder" int NOT NULL,
   "MeasureDimension" int NOT NULL,
   "Comment" text NULL
);
```

_AttributeFields

_AttributeFields describes the attributes of a dimension and their relational sources.

Column	Type	Description
DimensionName	String(50)	The name of the dimension.
TableId	Integer	The attribute table identifier. There are up to 3 allowable tables of attributes for a dimension. These are identified by a number 1, 2 or 3.
RelationalField	String(50)	The name of the relational column from which the attribute field is loaded.
FieldName	String(50)	The name of the attribute field in the OLAP database to be loaded with data.
FieldDescription	String(150)	The description of the field.
FieldType	String(1)	Enumerator indicating the field type. Values: C = Character string N = Numeric

Column	Type	Description
		D = Date L = Logical
FieldWidth	Integer	The width of the field.
FieldDecimalPlaces	Integer	The number of decimal places of a numeric field. For non-numeric fields this value must be NULL.
FieldOrderPosition	Integer	The order in which to load the attributes (within an attribute table). This number indicates the order position of an attribute (within its attribute table). A NULL value is assumed to be zero.

Comments

- The column DimensionName is linked to DimensionName of _Dimensions.
- The attributes are loaded when a dimension is loaded (_Jobs.JobType = 1) or when the attributes are loaded for an existing (already loaded) dimension (_Jobs.JobType = 6).
- If more than the maximum number of attribute fields is attempted to be loaded, then the OLAP dimension is created but loaded without any attribute tables.

_AttributeFields Table Creation with SQL Server

```
CREATE TABLE [dbo].[_AttributeFields](
   [DimensionName] [nvarchar](50) NOT NULL,
   [TableId] [int] NOT NULL,
   [RelationalField] [nvarchar](50) NOT NULL,
   [FieldName] [nvarchar](50) NOT NULL,
   [FieldDescription] [nvarchar](150) NULL,
   [FieldType] [nchar](1) NULL,
   [FieldWidth] [int] NOT NULL,
   [FieldDecimalPlaces] [int] NULL,
   [FieldOrderPosition] [int] NULL,
   [Comment] [ntext] NULL
)
```

_AttributeFields Table Creation with Oracle

```
CREATE TABLE "_AttributeFields"(
   "DimensionName" nvarchar2(50) NOT NULL,
   "TableId" int NOT NULL,
   "RelationalField" nvarchar2(50) NOT NULL,
   "FieldName" nvarchar2(50) NOT NULL,
   "FieldDescription" nvarchar2(150) NULL,
   "FieldType" nchar(1) NULL,
   "FieldWidth" int NOT NULL,
```

```
   "FieldDecimalPlaces" int NULL,
   "FieldOrderPosition" int NULL,
   "Comment" nclob NULL
);
```

_AttributesFields Table Creation with Postgres

```
CREATE TABLE "_AttributeFields"(
   "DimensionName" varchar(50) NOT NULL,
   "TableId" int NOT NULL,
   "RelationalField" varchar(50) NOT NULL,
   "FieldName" varchar(50) NOT NULL,
   "FieldDescription" varchar(150) NULL,
   "FieldType" char(1) NULL,
   "FieldWidth" int NOT NULL,
   "FieldDecimalPlaces" int NULL,
   "FieldOrderPosition" int NULL,
   "Comment" text NULL
);
```

_Subsets

_Subsets describes the subsets of the dimensions. A subset is an ordered set of elements contained within the set of elements of a dimension.

Column	Type	Description
DimName	String(50)	The name of the dimension.
SubsetName	String(50)	The name of the subset of the dimension.
SubsetKind	Integer	The type of the subset. Enumerator. Values: 1 = Element List 2 = Attribute Query (Not implemented) 3 = Data Query (Not implemented)
SubsetCategory	Integer	The category of subset. Enumerator. Values: 1 = Static 2 = Dynamic (Not implemented)

Column	Type	Description
SubsetTable	Text	The name of the relational table from which the subset is sourced. This can be a sub-query

Comments

- The column DimName is linked to DimensionName of _Dimensions.
- A dimension can have any number of subsets.
- The subsets are loaded when a dimension is loaded (_Jobs.JobType = 1) or when the subsets are loaded for an existing (already loaded) dimension (_Jobs.JobType = 8).

_Subsets Table Creation with SQL Server

```
CREATE TABLE [dbo].[_Subsets](
   [DimName] [nvarchar](50) NOT NULL,
   [SubsetName] [nvarchar](50) NOT NULL,
   [SubsetKind] [int] NOT NULL,
   [SubsetCategory] [int] NOT NULL,
   [SubsetTable] [ntext] NOT NULL,
   [Comment] [ntext] NULL
)
```

_Subsets Table Creation with Oracle

```
CREATE TABLE "_Subsets"(
   "DimName" nvarchar2(50) NOT NULL,
   "SubsetName" nvarchar2(50) NOT NULL,
   "SubsetKind" int NOT NULL,
   "SubsetCategory" int NOT NULL,
   "SubsetTable" nclob NOT NULL,
   "Comment" nclob NULL
);
```

_Subsets Table Creation with Postgres

```
CREATE TABLE "_Subsets"(
   "DimName" varchar(50) NOT NULL,
   "SubsetName" varchar(50) NOT NULL,
   "SubsetKind" int NOT NULL,
   "SubsetCategory" int NOT NULL,
   "SubsetTable" text NOT NULL,
   "Comment" text NULL
);
```

_CubeAccessControl

_CubeAccessControl relates fact cubes to their multi-dimensional access control cubes (their MDACs).

Column	Type	Description
CubeName	String(50)	The name of the cube whose data access cube to relate.
AccessCubeName	String(50)	The name of an access control cube of the cube named in CubeName.

Comments

- The column CubeName is linked from _Cubes.CubeName.
- One fact cube can have many access control cubes.

_CubeAccessControl Table Creation with SQL Server

```
CREATE TABLE [dbo].[_CubeAccessControl](
   [CubeName] [nvarchar](50) NOT NULL,
   [AccessCubeName] [nvarchar](50) NOT NULL,
   [Comment] [ntext] NULL
)
```

_CubeAccessControl Table Creation with Oracle

```
CREATE TABLE "_CubeAccessControl"(
   "CubeName" nvarchar2(50) NOT NULL,
   "AccessCubeName" nvarchar2(50) NOT NULL,
   "Comment" nclob NULL
);
```

_CubeAccessControl Table Creation with Postgres

```
CREATE TABLE "_CubeAccessControl"(
   "CubeName" varchar(50) NOT NULL,
   "AccessCubeName" varchar(50) NOT NULL,
   "Comment" text NULL
);
```

_JobsParameters

_JobsParameters describes the element filters for partial fact loads.

Column	Type	Description
JobParameter	BigInteger	Identifies a set of element filters for a partial load. All parameter sets used in a partial load have the same JobParameter.
ParameterSet	Integer	Integer identifying the set of dimensions defining one filter for a load. All dimensions relating to one filter must have the same parameter set, but this can otherwise be arbitrary.
DimIdx	Integer	The index of the dimension within the cube being restricted (to the list of elements in the column ParameterValue). The index must be zero based.
ParameterValue	Text	Comma-separated list of elements of the dimension whose index is DimIdx to which the loading is restricted for the related job (linked through JobParameter). **Note:** Element names containing commas must be enclosed either in single quotes or double quotes. Single quotes would be used to enclose element names containing double quotes and vice versa. Both single and double quotes cannot be used together in the same element name. (If this is done, the behavior of the process is undefined and will give unexpected results.)

Comments

- The value of JobParameter is linked through the JobParameter column of the _Jobs table.
- For this type of load to be allowed, _FactLoad.FactLoadType must be 2 (indicating partial load. Where this is not 2, _Jobs.JobParameter must be NULL and no corresponding rows on _JobsParameters are allowed).
- A set of dimensions defining a filter must have the same ParameterSet.
- Dimensions of the cube omitted from a parameter set represent unfiltered dimensions. Facts in the cube relating to all elements of these dimensions are loaded.
- Omitting the dimension from a parameter set includes all elements in that dimension.

- The set of parameters for one value of ParameterSet define one filter. These define the coordinates of the data to load through element names that are to be selected from the table. But what the OLAP Server does first is to clear a space in the OLAP database of any data occupying those coordinate positions. Moreover, the data to load need not necessarily fill the entire space cleared away. Indeed, it need not fill any of it. So these parameters can be used as a means to delete data from the OLAP database: simply use an empty table with related _JobsParameter data defining the elements coordinates of the data to be cleared away.
- The space cleared away may be further restricted by fact load parameters. (See _FactLoadParameters.)
- An element job parameter could be qualified with the name of the hierarchy it belongs to. If the element data in the fact table uses hierarchy names, so must the matching job parameters.
  - The two names are separated with a tab character. Hierarchy name appears first: heirarchyName<tab>elementName.
- You can include consolidated elements by prefixing the element name with a "B". The elements are expanded in the processing to their base level descendants. For example, March,B:'2nd Quarter',July expands to March,April,May,Jun,July.

_JobsParameters Table Creation with SQL Server

```
CREATE TABLE [dbo].[_JobsParameters](
   [JobParameter] [bigint] NOT NULL,
   [ParameterSet] [int] NOT NULL,
   [DimIdx] [int] NOT NULL,
   [ParameterValue] [ntext] NOT NULL,
   [Comment] [ntext] NULL
)
```

_JobsParameters Table Creation with Oracle

```
CREATE TABLE "_JobsParameters"(
   "JobParameter" number(19, 0) NOT NULL,
   "ParameterSet" int NOT NULL,
   "DimIdx" int NOT NULL,
   "ParameterValue" nclob NOT NULL,
   "Comment" nclob NULL
);
```

_JobsParameters Table Creation with Postgres

```
CREATE TABLE "_JobsParameters"(
   "JobParameter" bigint NOT NULL,
   "ParameterSet" int NOT NULL,
   "DimIdx" int NOT NULL,
   "ParameterValue" text NOT NULL,
   "Comment" text NULL
);
```

_FactLoad

_FactLoad describes the loading of facts at cube level. Only used by job types 3 (Load Cube) and 9 (Clear Cube Region).

Column	Type	Description
JobObject	String(50)	The job object of the fact load. This identifies the load of a cube from a specific table.
CubeId	String(50)	The name of the cube to load (with facts).
FactTable	Text	The name of the table to load the cube from. The cube's name is in CubeId. A sub-query can be used in place of a table name. Must not be NULL for a fact load. Must be NULL for clearing a cube region.
FactChangesTable	Text	For incremental load only (FactLoadType 3). The name of the table containing version numbers and their related coordinate/key data for processing. The records of this table selected are in the range defined by the values in the fields LastLoadedVersion and NewestVersionToLoad of this table. The corresponding key data link to the fact table for the selection of the fact data with which to update (overwrite) the matching cells in the cube. The use of a fact changes table is optional. It will only be used if version numbers will not be present in the fact table. If a fact changes table is not used, the value of this field must be NULL. A sub-query can be used in place of a table name.
FactLoadType	Integer	The fact load type. Enumerator, values: 1 = Full Load. 2 = Partial Load. 3 = Incremental Load **Note:** Fact Load Type 3 (Incremental Load) is not allowed for Job Type 9 (Clear Cube Region).
ParameterId	Integer	The parameter id. This is the key to dimension-level loading criteria described in _FactLoadParameters.
LastLoadedVersion	Integer	Version number of last incremental load. Next incremental load beginning from version number is one more than this. Must be NULL for loads other than incremental. Must not be NULL for incremental.
NewestVersionToLoad	Integer	Load all versions up to and including this version number. Must be NULL for loads other than incremental. Must not be NULL for incremental.

Column	Type	Description
SourceId	Integer	Data source of cube. This links to the key SourceId of the table _Sources which describes the details of the source. NULL means the cube source uses the same connection as in `Db.ini`.
FactType	Integer	The data type of the facts described by the current fact load (found in the facts column of the fact table). Enumerator, values: 1 = Numeric 2 = Character String 3 = Numeric without summation/aggregation Further notes: Must not be NULL for Fact Load Type 3 (Load Cube) Must be NULL for Fact Load Type 9 (Clear Cube Region)
RelationalColumnName	String(50)	The name of the relational column in the fact table describing the facts. Further notes: Can be NULL for Fact Load Type 3 (Load Cube) Must be NULL for Fact Load Type 9 (Clear Cube Region)
LoadFlags	BigInteger	Flags controlling the fact load. The following powers of 2 flag the action indicated: 1 = Do not break the import on the first unresolved error that occurs. 2 = Document each error that occurs. 4 = Commit the imported values even if unresolved errors have occurred. 32= Import zero values. 8 = Document replaced missing and unknown elements even if these resulted in successful writing of data to the cube. • Combinations of these can be made by adding their values together. • Committing data and breaking on error are incompatible. See comments. • A commit will occur regardless if there are no errors in the load. • Default value is 0. A NULL value is interpreted as 0.

Comments

- Only used by job types 3 (Load Cube) and 9 (Clear Cube Region).
- A fact load is defined by a source-table-target-cube pair. This metadata table defines the fact loads.
- JobObject identifies a fact load.
- The column JobObject is linked from the _Jobs table column of the same name.
- Fact data of type 1 (numeric) is aggregated over all dimension columns with ‘group by’ and ‘sum’ to easily handle fact tables with more detail than is loaded into the cube. This is (obviously) not done for fact data of type 2 (string), as there is no viable default aggregation available there and it is also omitted for fact type 3 (numeric without aggregation), mainly for performance reasons when querying large data-sets that are already pre-aggregated.
- The complete picture of a fact load is described by this table together with the _FactLoadParameters table.
- For incremental fact loads, the field LastLoadedVersion is updated at the end of the run with the NewestVersionToLoad unless this is -1. If NewestVersionToLoad is -1 the field LastLoadedVersion is updated at the end of the run with the highest version number actually loaded. The field NewestVersionToLoad has to be updated externally.
- For incremental loads there are two scenarios, one where the versions are stored in the rows of the fact table, and the other in the rows of a separate fact changes table.
  - Without fact changes table: additional rows coming in are considered to be deltas, changes to the already existing values. These rows are entered into the cube additively by selected version. The values of loaded data are added to values of existing data with matching key coordinates.
  - With fact changes table: rows in the fact changes table are considered to mark cells with changed values. Matching existing data is overwritten by the new loaded values. Rows in the fact changes table are selected by version number, and these are linked by key to the corresponding rows in the fact table. All rows of the fact table whose keys match will be selected.
- If RelationalColumnName is NULL, then ColumnName and TargetElement of _FactLoadParameters must both be set in at least one row. If it is not NULL, then these cannot both be set in any row.
- To commit valid data even if errors have occurred, the 4-bit and the 1-bit must be set. Any value of LoadFlags that entails the 4-bit being set but not the 1-bit will result in an error.

_FactLoad Table Creation with SQL Server

```
CREATE TABLE [dbo].[_FactLoad](
   [JobObject] [nvarchar](50) NOT NULL UNIQUE,
   [CubeId] [nvarchar](50) NOT NULL,
   [FactTable] [ntext] NULL,
   [FactChangesTable] [ntext] NULL,
   [FactLoadType] [int] NOT NULL,
   [ParameterId] [int] NOT NULL,
   [LastLoadedVersion] [int] NULL,
   [NewestVersionToLoad] [int] NULL,
   [SourceId] [int] NULL,
   [FactType] [int] NULL,
   [RelationalColumnName] [nvarchar](50) NULL,
   [LoadFlags] [bigint] NULL,
```

```
   [Comment] [ntext] NULL
)
```

_FactLoad Table Creation with Oracle

```
CREATE TABLE "_FactLoad"(
   "JobObject" nvarchar2(50) NOT NULL UNIQUE,
   "CubeId" nvarchar2(50) NOT NULL,
   "FactTable" nclob NULL,
   "FactChangesTable" nclob NULL,
   "FactLoadType" int NOT NULL,
   "ParameterId" int NOT NULL,
   "LastLoadedVersion" int NULL,
   "NewestVersionToLoad" int NULL,
   "SourceId" int NULL,
   "FactType" int NULL,
   "RelationalColumnName" nvarchar2(50) NULL,
   "LoadFlags" number(19, 0) NULL,
   "Comment" nclob NULL
);
```

_FactLoad Table Creation with Postgres

```
CREATE TABLE "_FactLoad"(
   "JobObject" varchar(50) NOT NULL UNIQUE,
   "CubeId" varchar(50) NOT NULL,
   "FactTable" text NULL,
   "FactChangesTable" text NULL,
   "FactLoadType" int NOT NULL,
   "ParameterId" int NOT NULL,
   "LastLoadedVersion" int NULL,
   "NewestVersionToLoad" int NULL,
   "SourceId" int NULL,
   "FactType" int NULL,
   "RelationalColumnName" varchar(50) NULL,
   "LoadFlags" bigint NULL,
   "Comment" text NULL
);
```

_FactLoadParameters

_FactLoadParameters describes a fact load at dimension level. Only used by job types 3 (Load Cube) and 9 (Clear Cube Region).

Column	Type	Description
Id	Integer	The ID of the fact load. This is the ParameterId on the _FactLoad table.
DimIdx	Integer	The index of the dimension of the target cube of the fact load relating to the parameter in the column ID. The index must be zero based.
ColumnName	String(50)	The name of the source table column matching the dimension, if the dimension of the target cube relates to a table column. This is NULL if the whole source table relates to a single element of the dimension in DimIdx of the target cube.
TargetElement	String(71)	The name of the target element of the source table of the dimension DimIdx if this does not correspond to a column of the source table. This is NULL if the dimension with index DimIdx relates to a column of the source table.
TargetElementUnknown	String(71)	If the element name cannot be found in the dimension, the process should use this element name instead. This works no matter if the element name was delivered as a TargetElement or in a column.
TargetElementMissing	String(71)	If the element name is empty, the process should use this element name instead. This works no matter if the element name was delivered as a TargetElement or in a column.

Comments

- Only used by job types 3 (Load Cube) and 9 (Clear Cube Region).
- _FactLoadParameters relates the dimensions of a target cube with characteristics of its source table.
- For each dimension of a target cube, it relates exactly one of the following:
  - that dimension of the target cube to a column of a source table,
  - the entire table to an element of that dimension of the target cube.

- ColumnName and TargetElement cannot both be NULL. They can both be non-NULL. If they are both non-NULL, the column name refers to a value column of the fact table and corresponds to an element of a dimension. There can be any number of rows with both ColumnName and TargetElement set, but they can only relate to one dimension.
- When a full fact load occurs, a region in the target cube defined by the dimension indices and target elements is cleared away first before loading takes place (to this region). Note that the loading might not fill the cleared region entirely (or indeed at all).
- When a partial fact load occurs, a region in the target cube defined by:
  - the dimension indices and target elements
  - restricted further by jobs parameters defining the partial load

  is cleared away first before loading takes place (to this region). Note that the loading might not fill the cleared region entirely (or indeed at all).
- Target, unknown and missing elements could be qualified with the name of the hierarchy each belongs to.
  - The two names are separated with a tab character. Hierarchy name appears first: hierarchyName<tab>elementName.

_FactLoadParameters Table Creation with SQL Server

```
CREATE TABLE [dbo].[_FactLoadParameters](
   [Id] [int] NOT NULL,
   [DimIdx] [int] NOT NULL,
   [ColumnName] [nvarchar](50) NULL,
   [TargetElement] [nvarchar](71) NULL,
   [TargetElementUnknown] [nvarchar](71) NULL,
   [TargetElementMissing] [nvarchar](71) NULL,
   [Comment] [ntext] NULL
)
```

_FactLoadParameters Table Creation with Oracle

```
CREATE TABLE "_FactLoadParameters"(
   "Id" int NOT NULL,
   "DimIdx" int NOT NULL,
   "ColumnName" nvarchar2(50) NULL,
   "TargetElement" nvarchar2(71) NULL,
   "TargetElementUnknown" nvarchar2(71) NULL,
   "TargetElementMissing" nvarchar2(71) NULL,
   "Comment" nclob NULL
);
```

_FactLoadParameters Table Creation with Postgres

```
CREATE TABLE "_FactLoadParameters"(
   "Id" int NOT NULL,
   "DimIdx" int NOT NULL,
```

```
    "ColumnName" varchar(50) NULL,
    "TargetElement" varchar(71) NULL,
    "TargetElementUnknown" varchar(71) NULL,
    "TargetElementMissing" varchar(71) NULL,
    "Comment" text NULL
);
```

_Hierarchies

_Hierarchies describes the hierarchies to load.

Column	Type	Description
HierarchyName	String(50)	Name of hierarchy
HierarchyDescrip-tion	String(150)	Description of hierarchy
DimensionName	String(50)	Name of dimension hierarchy belongs to.
SourceId	Integer	Data source of hierarchy. This links to the key SourceId of the table _Sources which describes the details of the source. NULL means the hierarchy source uses the same connection as in `Db.ini`.
HierarchyType	Integer	Enumerator for the hierarchy type of the hierarchy. 1 = Parent/Child relationship 2 = Attribute Driven Hierarchy
Param1	Text	First parameter. For HierarchyType = 1, this is the relational table source of the hierarchy data, the members' table. This can be a sub-query. For HierarchyType = 2, this is the Id of the hierarchy parameter list.
Param2	Text	Second parameter. For HierarchyType = 1, this is the relational table source of the hierarchy relations data. This might be the same table as the members' table found in Param1. NULL if there are no relations for the relational source. This can be a sub-query.
DefaultElement	String(71)	If an element of a hierarchy is not specified, take it to be this one. NULL if there is no default element. If specified default element is not present in hierarchy, no default element property is created and a warning is issued in the log file.

Column	Type	Description
ExtendedProps	Text	The extended properties of the hierarchy expressed as an XML document string. NULL means the hierarchy has no extended properties.

Comments

Param1 for HierarchyType 2 is the numeric Id of the list of parameters describing the Attribute Driven Hierarchies found in the _Parameters table.

_Hierarchies Table Creation with SQL Server

```
CREATE TABLE [dbo].[_Hierarchies](
   [HierarchyName] [nvarchar](50) NOT NULL,
   [HierarchyDescription] [nvarchar](150) NOT NULL,
   [DimensionName] [nvarchar](50) NOT NULL,
   [SourceId] [int] NULL,
   [HierarchyType] [int] NOT NULL,
   [Param1] [ntext] NULL,
   [Param2] [ntext] NULL,
   [DefaultElement] [nvarchar](71) NULL,
   [ExtendedProps] [ntext] NULL,
   [Comment] [ntext] NULL
)
```

_Hierarchies Table Creation with Oracle

```
CREATE TABLE "_Hierarchies"(
   "HierarchyName" nvarchar2(50) NOT NULL,
   "HierarchyDescription" nvarchar2(150) NOT NULL,
   "DimensionName" nvarchar2(50) NOT NULL,
   "SourceId" int NULL,
   "HierarchyType" int NOT NULL,
   "Param1" nclob NULL,
   "Param2" nclob NULL,
   "DefaultElement" nvarchar2(71) NULL,
   "ExtendedProps" nclob NULL,
   "Comment" nclob NULL
);
```

_Hierarchies Table Creation with Postgres

```
CREATE TABLE "_Hierarchies"(
   "HierarchyName" varchar(50) NOT NULL,
   "HierarchyDescription" varchar(150) NOT NULL,
   "DimensionName" varchar(50) NOT NULL,
   "SourceId" int NULL,
   "HierarchyType" int NOT NULL,
   "Param1" text NULL,
```

```
   "Param2" text NULL,
   "DefaultElement" varchar(71) NULL,
   "ExtendedProps" text NULL,
   "Comment" text NULL
);
```

_Parameters

_Parameters describes general parameters used by other parts of the system.

Column	Type	Description
Id	BigInteger	The ID of the parameter.
Key	String(50)	The key of the parameter.
DataType	Integer	Enumerator for the data type of the parameter: 0 = Link Id 1 = String 2 = Integer
StringValue	Text	The value of a string parameter. This is NULL if the parameter is of another type.
IntegerValue	Integer	The value of an integer parameter. This is NULL if the parameter is of another type.
LinkId	BigInteger	The Id of the rows this row links to. This is NULL if the parameter is of another type.

Comments

The pair (Id, Key) makes a primary key

_Parameters Table Creation with SQL Server

```
CREATE TABLE [dbo].[_Parameters](
   [Id] [bigint] NOT NULL,
   [Key] [nvarchar](50) NOT NULL,
   [DataType] [int] NOT NULL,
   [StringValue] [ntext] NULL,
   [IntegerValue] [int] NULL,
   [LinkId] [bigint] NULL,
   [Comment] [ntext] NULL
   CONSTRAINT ID_KEY_PK PRIMARY KEY ([Id], [Key])
)
```

_Parameters Table Creation with Oracle

```
CREATE TABLE "_Parameters"(
   "Id" number(38, 0) NOT NULL,
   "Key" nvarchar2(50) NOT NULL,
   "DataType" int NOT NULL,
   "StringValue" nclob NULL,
   "IntegerValue" int NULL,
   "LinkId" number(38, 0) NULL,
   "Comment" nclob NULL,
   CONSTRAINT ID_KEY_PK PRIMARY KEY ("Id", "Key")
);
```

_Parameters Table Creation with Postgres

```
CREATE TABLE "_Parameters"(
   "Id" bigint NOT NULL,
   "Key" varchar(50) NOT NULL,
   "DataType" int NOT NULL,
   "StringValue" text NULL,
   "IntegerValue" int NULL,
   "LinkId" bigint NULL,
   "Comment" text NULL,
   CONSTRAINT ID_KEY_PK PRIMARY KEY ("Id", "Key")
);
```

_ErrorText

_ErrorText lists the error texts for each Olap error code in each available language/region.

Column	Type	Description
ErrorCode	Integer	Olap error code.
LanguageRegion	String(15)	Language/region. See comments.
Text	Text	Error text.

Comments

- · LanguageRegion conforms to W3C standards. See http://www.w3.org/International/articles/language-tags/.
- · LanguageRegion is composed of two components separated by a hyphen "-":
  - Language code
  - Region code

_ErrorText Table Creation with SQL Server

```
CREATE TABLE [dbo].[_ErrorText](
   [ErrorCode] [int] NOT NULL,
   [LanguageRegion] [nvarchar](15) NOT NULL,
   [Text] [ntext] NULL
)
```

_ErrorText Table Creation with Oracle

```
CREATE TABLE "_ErrorText"(
   "ErrorCode" int NOT NULL,
   "LanguageRegion" nvarchar2(15) NOT NULL,
   "Text" nclob NULL
);
```

_ErrorText Table Creation with Postgres

```
CREATE TABLE "_ErrorText"(
   "ErrorCode" int NOT NULL,
   "LanguageRegion" varchar(15) NOT NULL,
   "Text" text NULL
);
```

_ErrorLogFactLoad

_ErrorLogFactLoad logs fact load errors of operations on cubes.

Column	Type	Description
JobId	BigInteger	The ID of a job which has an error.
ErrorCode	Integer	Olap error code.
ErrorDimIdx	Integer	Index of the dimension whose element-coordinate is causing this error. NULL if not relevant or available.
ElementNN	String(256)	Element in the NNth coordinate position. As many elements as there are dimensions in the cube. The rest are NULL.
ElementCap-tionNN	String(256)	Caption of element in the NNth coordinate position. As many as there are elements. NULL if not available.
FactType	Integer	Enumerator. • · 1 = Numeric • · 2 = String NULL for ClearCubeRegion.

Column	Type	Description
ValueNumeric	Real	Numeric value attempted to write to cell. Might be NULL.
ValueString	Text	String value attempted to write to cell. Might be NULL.

Comments

- ElementNN and ElementCaptionNN are 0-based coordinate positions of dimensions within the cube. The elements begin with Element00 and end with a possible Element29. The base of 0 reflects how the Olap server indexes dimensions within the cube. There are a maximum of 30 allowable dimensions in the present Olap version.
- FactType on this table is not quite the same as on FactType on _FactLoadParameter, which also has a third option for Numeric No Sum. Here, no distinction is made between this and 1 (Numeric).

_ErrorLogFactLoad Table Creation with SQL Server

```
CREATE TABLE [dbo].[_ErrorLogFactLoad](
   [JobId] [bigint] NOT NULL,
   [ErrorCode] [int] NOT NULL,
   [ErrorDimIdx] [int] NULL,
   [ErrorDimName] [nvarchar](50) NULL,
   [Element00] [nvarchar](256) NULL,
   [Element01] [nvarchar](256) NULL,
   [Element02] [nvarchar](256) NULL,
   [Element03] [nvarchar](256) NULL,
   [Element04] [nvarchar](256) NULL,
   [Element05] [nvarchar](256) NULL,
   [Element06] [nvarchar](256) NULL,
   [Element07] [nvarchar](256) NULL,
   [Element08] [nvarchar](256) NULL,
   [Element09] [nvarchar](256) NULL,
   [Element10] [nvarchar](256) NULL,
   [Element11] [nvarchar](256) NULL,
   [Element12] [nvarchar](256) NULL,
   [Element13] [nvarchar](256) NULL,
   [Element14] [nvarchar](256) NULL,
   [Element15] [nvarchar](256) NULL,
   [Element16] [nvarchar](256) NULL,
   [Element17] [nvarchar](256) NULL,
   [Element18] [nvarchar](256) NULL,
   [Element19] [nvarchar](256) NULL,
   [Element20] [nvarchar](256) NULL,
   [Element21] [nvarchar](256) NULL,
   [Element22] [nvarchar](256) NULL,
   [Element23] [nvarchar](256) NULL,
   [Element24] [nvarchar](256) NULL,
   [Element25] [nvarchar](256) NULL,
   [Element26] [nvarchar](256) NULL,
   [Element27] [nvarchar](256) NULL,
   [Element28] [nvarchar](256) NULL,
   [Element29] [nvarchar](256) NULL,
```

```
   [ElementCaption00] [nvarchar](256) NULL,
   [ElementCaption01] [nvarchar](256) NULL,
   [ElementCaption02] [nvarchar](256) NULL,
   [ElementCaption03] [nvarchar](256) NULL,
   [ElementCaption04] [nvarchar](256) NULL,
   [ElementCaption05] [nvarchar](256) NULL,
   [ElementCaption06] [nvarchar](256) NULL,
   [ElementCaption07] [nvarchar](256) NULL,
   [ElementCaption08] [nvarchar](256) NULL,
   [ElementCaption09] [nvarchar](256) NULL,
   [ElementCaption10] [nvarchar](256) NULL,
   [ElementCaption11] [nvarchar](256) NULL,
   [ElementCaption12] [nvarchar](256) NULL,
   [ElementCaption13] [nvarchar](256) NULL,
   [ElementCaption14] [nvarchar](256) NULL,
   [ElementCaption15] [nvarchar](256) NULL,
   [ElementCaption16] [nvarchar](256) NULL,
   [ElementCaption17] [nvarchar](256) NULL,
   [ElementCaption18] [nvarchar](256) NULL,
   [ElementCaption19] [nvarchar](256) NULL,
   [ElementCaption20] [nvarchar](256) NULL,
   [ElementCaption21] [nvarchar](256) NULL,
   [ElementCaption22] [nvarchar](256) NULL,
   [ElementCaption23] [nvarchar](256) NULL,
   [ElementCaption24] [nvarchar](256) NULL,
   [ElementCaption25] [nvarchar](256) NULL,
   [ElementCaption26] [nvarchar](256) NULL,
   [ElementCaption27] [nvarchar](256) NULL,
   [ElementCaption28] [nvarchar](256) NULL,
   [ElementCaption29] [nvarchar](256) NULL,
   [FactType] [int] NULL,
   [ValueNumeric] [real] NULL,
   [ValueString] [ntext] NULL,
   [Comment] [ntext] NULL
)
```

_ErrorLogFactLoad Table Creation with Oracle

```
CREATE TABLE "_ErrorLogFactLoad"(
   "JobId" number(19, 0) NOT NULL,
   "ErrorCode" int NOT NULL,
   "ErrorDimIdx" int NULL,
   "ErrorDimName" nvarchar2(50) NULL,
   "Element00" nvarchar2(256) NULL,
   "Element01" nvarchar2(256) NULL,
   "Element02" nvarchar2(256) NULL,
   "Element03" nvarchar2(256) NULL,
   "Element04" nvarchar2(256) NULL,
   "Element05" nvarchar2(256) NULL,
   "Element06" nvarchar2(256) NULL,
   "Element07" nvarchar2(256) NULL,
```

```
    "Element08" nvarchar2(256) NULL,
    "Element09" nvarchar2(256) NULL,
    "Element10" nvarchar2(256) NULL,
    "Element11" nvarchar2(256) NULL,
    "Element12" nvarchar2(256) NULL,
    "Element13" nvarchar2(256) NULL,
    "Element14" nvarchar2(256) NULL,
    "Element15" nvarchar2(256) NULL,
    "Element16" nvarchar2(256) NULL,
    "Element17" nvarchar2(256) NULL,
    "Element18" nvarchar2(256) NULL,
    "Element19" nvarchar2(256) NULL,
    "Element20" nvarchar2(256) NULL,
    "Element21" nvarchar2(256) NULL,
    "Element22" nvarchar2(256) NULL,
    "Element23" nvarchar2(256) NULL,
    "Element24" nvarchar2(256) NULL,
    "Element25" nvarchar2(256) NULL,
    "Element26" nvarchar2(256) NULL,
    "Element27" nvarchar2(256) NULL,
    "Element28" nvarchar2(256) NULL,
    "Element29" nvarchar2(256) NULL,
    "ElementCaption00" nvarchar2(256) NULL,
    "ElementCaption01" nvarchar2(256) NULL,
    "ElementCaption02" nvarchar2(256) NULL,
    "ElementCaption03" nvarchar2(256) NULL,
    "ElementCaption04" nvarchar2(256) NULL,
    "ElementCaption05" nvarchar2(256) NULL,
    "ElementCaption06" nvarchar2(256) NULL,
    "ElementCaption07" nvarchar2(256) NULL,
    "ElementCaption08" nvarchar2(256) NULL,
    "ElementCaption09" nvarchar2(256) NULL,
    "ElementCaption10" nvarchar2(256) NULL,
    "ElementCaption11" nvarchar2(256) NULL,
    "ElementCaption12" nvarchar2(256) NULL,
    "ElementCaption13" nvarchar2(256) NULL,
    "ElementCaption14" nvarchar2(256) NULL,
    "ElementCaption15" nvarchar2(256) NULL,
    "ElementCaption16" nvarchar2(256) NULL,
    "ElementCaption17" nvarchar2(256) NULL,
    "ElementCaption18" nvarchar2(256) NULL,
    "ElementCaption19" nvarchar2(256) NULL,
    "ElementCaption20" nvarchar2(256) NULL,
    "ElementCaption21" nvarchar2(256) NULL,
    "ElementCaption22" nvarchar2(256) NULL,
    "ElementCaption23" nvarchar2(256) NULL,
    "ElementCaption24" nvarchar2(256) NULL,
    "ElementCaption25" nvarchar2(256) NULL,
    "ElementCaption26" nvarchar2(256) NULL,
    "ElementCaption27" nvarchar2(256) NULL,
    "ElementCaption28" nvarchar2(256) NULL,
    "ElementCaption29" nvarchar2(256) NULL,
    "FactType" int NULL,
```

```
   "ValueNumeric" real NULL,
   "ValueString" nclob NULL,
   "Comment" nclob NULL
);
```

_ErrorLogFactLoad Table Creation with Postgres

```
CREATE TABLE "_ErrorLogFactLoad"(
   "JobId" bigint NOT NULL,
   "ErrorCode" int NOT NULL,
   "ErrorDimIdx" int NULL,
   "ErrorDimName" varchar(50) NULL,
   "Element00" varchar(256) NULL,
   "Element01" varchar(256) NULL,
   "Element02" varchar(256) NULL,
   "Element03" varchar(256) NULL,
   "Element04" varchar(256) NULL,
   "Element05" varchar(256) NULL,
   "Element06" varchar(256) NULL,
   "Element07" varchar(256) NULL,
   "Element08" varchar(256) NULL,
   "Element09" varchar(256) NULL,
   "Element10" varchar(256) NULL,
   "Element11" varchar(256) NULL,
   "Element12" varchar(256) NULL,
   "Element13" varchar(256) NULL,
   "Element14" varchar(256) NULL,
   "Element15" varchar(256) NULL,
   "Element16" varchar(256) NULL,
   "Element17" varchar(256) NULL,
   "Element18" varchar(256) NULL,
   "Element19" varchar(256) NULL,
   "Element20" varchar(256) NULL,
   "Element21" varchar(256) NULL,
   "Element22" varchar(256) NULL,
   "Element23" varchar(256) NULL,
   "Element24" varchar(256) NULL,
   "Element25" varchar(256) NULL,
   "Element26" varchar(256) NULL,
   "Element27" varchar(256) NULL,
   "Element28" varchar(256) NULL,
   "Element29" varchar(256) NULL,
   "ElementCaption00" varchar(256) NULL,
   "ElementCaption01" varchar(256) NULL,
   "ElementCaption02" varchar(256) NULL,
   "ElementCaption03" varchar(256) NULL,
   "ElementCaption04" varchar(256) NULL,
   "ElementCaption05" varchar(256) NULL,
   "ElementCaption06" varchar(256) NULL,
   "ElementCaption07" varchar(256) NULL,
   "ElementCaption08" varchar(256) NULL,
```

```
    "ElementCaption09" varchar(256) NULL,
    "ElementCaption10" varchar(256) NULL,
    "ElementCaption11" varchar(256) NULL,
    "ElementCaption12" varchar(256) NULL,
    "ElementCaption13" varchar(256) NULL,
    "ElementCaption14" varchar(256) NULL,
    "ElementCaption15" varchar(256) NULL,
    "ElementCaption16" varchar(256) NULL,
    "ElementCaption17" varchar(256) NULL,
    "ElementCaption18" varchar(256) NULL,
    "ElementCaption19" varchar(256) NULL,
    "ElementCaption20" varchar(256) NULL,
    "ElementCaption21" varchar(256) NULL,
    "ElementCaption22" varchar(256) NULL,
    "ElementCaption23" varchar(256) NULL,
    "ElementCaption24" varchar(256) NULL,
    "ElementCaption25" varchar(256) NULL,
    "ElementCaption26" varchar(256) NULL,
    "ElementCaption27" varchar(256) NULL,
    "ElementCaption28" varchar(256) NULL,
    "ElementCaption29" varchar(256) NULL,
    "FactType" int NULL,
    "ValueNumeric" real NULL,
    "ValueString" text NULL,
    "Comment" text NULL
);
```

_ErrorLogElementLoad

_ ErrorLogElementLoad logs errors in element loads.

Column	Type	Description
JobId	BigInteger	The ID of a job which has an error.
ErrorCode	Integer	Olap error code.
Dimension	String(256)	Dimension name.
DimensionCap-tion	String(256)	Dimension caption.
Hierarchy	String(256)	Hierarchy name.
HierarchyCaption	String(256)	Hierarchy caption.
Element	String(256)	Element or child element name.
ElementCaption	String(256)	Element or child element caption.
ElementType	String(256)	Element or child element type.

Column	Type	Description
ParentElement	String(256)	Parent element name. NULL if the occurred during the load of the member table.
ParentElement-Caption	String(256)	Parent element caption. NULL if the error occurred during the load of the member table or if the caption is not available.
Weight	Real	The weight of the element. NULL if the error occurred during the load of the member table or if the caption is not available.

_ErrorLogElementLoad Table Creation with SQL Server

```
CREATE TABLE [dbo].[_ErrorLogElementLoad](
   [JobId] [bigint] NOT NULL,
   [ErrorCode] [int] NOT NULL,
   [Dimension] [nvarchar](256) NULL,
   [DimensionCaption] [nvarchar](256) NULL,
   [Hierarchy] [nvarchar](256) NULL,
   [HierarchyCaption] [nvarchar](256) NULL,
   [Element] [nvarchar](256) NULL,
   [ElementCaption] [nvarchar](256) NULL,
   [ElementType] [nvarchar](256) NULL,
   [ParentElement] [nvarchar](256) NULL,
   [ParentElementCaption] [nvarchar](256) NULL,
   [Weight] [real] NULL,
   [Comment] [ntext] NULL
)
```

_ErrorLogElementLoad Table Creation with Oracle

```
CREATE TABLE "_ErrorLogElementLoad"(
   "JobId" number(19, 0) NOT NULL,
   "ErrorCode" int NOT NULL,
   "Dimension" nvarchar2(256) NULL,
   "DimensionCaption" nvarchar2(256) NULL,
   "Hierarchy" nvarchar2(256) NULL,
   "HierarchyCaption" nvarchar2(256) NULL,
   "Element" nvarchar2(256) NULL,
   "ElementCaption" nvarchar2(256) NULL,
   "ElementType" nvarchar2(256) NULL,
   "ParentElement" nvarchar2(256) NULL,
   "ParentElementCaption" nvarchar2(256) NULL,
   "Weight" real NULL,
   "Comment" nclob NULL
);
```

_ErrorLogElementLoad Table Creation with Postgres

```
CREATE TABLE "_ErrorLogElementLoad"(
   "JobId" bigint NOT NULL,
   "ErrorCode" int NOT NULL,
   "Dimension" varchar(256) NULL,
   "DimensionCaption" varchar(256) NULL,
   "Hierarchy" varchar(256) NULL,
   "HierarchyCaption" varchar(256) NULL,
   "Element" varchar(256) NULL,
   "ElementCaption" varchar(256) NULL,
   "ElementType" varchar(256) NULL,
   "ParentElement" varchar(256) NULL,
   "ParentElementCaption" varchar(256) NULL,
   "Weight" real NULL,
   "Comment" text NULL
);
```

_ErrorLogOther

_ErrorText lists the error texts for each Olap error code in each available language/region.

Column	Type	Description
JobId	BigInteger	The id of a job which has an error.
ErrorCode	Integer	Olap error code.
ObjectKind	String(256)	The error object kind.
ObjectName	String(256)	The name of the error object.
ObjectCaption	String(256)	The caption of the error object. If meaningful and if available in the process.
ErrorText	Text	Human-readable description of error.

Comments

- ObjectKind refers to the type of object the row describes. Examples are CUBE, DIMENSION, ELEMENT, JOB_OBJECT.

_ErrorLogOther Table Creation with SQL Server

```
CREATE TABLE [dbo].[_ErrorLogOther](
   [JobId] [bigint] NOT NULL,
   [ErrorCode] [int] NOT NULL,
   [ObjectKind] [nvarchar](256) NULL,
   [ObjectName] [nvarchar](256) NULL,
   [ObjectCaption] [nvarchar](256) NULL,
```

```
   [ErrorText] [text] NULL
)
```

_ErrorLogOther Table Creation with Oracle

```
CREATE TABLE "_ErrorLogOther"(
   "JobId" number(19, 0) NOT NULL,
   "ErrorCode" int NOT NULL,
   "ObjectKind" nvarchar2(256) NULL,
   "ObjectName" nvarchar2(256) NULL,
   "ObjectCaption" nvarchar2(256) NULL,
   "ErrorText" nclob NULL
);
```

_ErrorLogOther Table Creation with Postgres

```
CREATE TABLE "_ErrorLogOther"(
   "JobId" bigint NOT NULL,
   "ErrorCode" int NOT NULL,
   "ObjectKind" varchar(256) NULL,
   "ObjectName" varchar(256) NULL,
   "ObjectCaption" varchar(256) NULL,
   "ErrorText" text NULL
);
```

Data tables

There are four kinds of data tables:

1. Member tables (or element tables)
2. Relationship tables (or hierarchy tables)
3. Subset tables
4. Fact tables.

There can be any number of each of these and they can have any names. These names are defined by the user but they must be stored in the appropriate places in the metadata tables so the system knows where and how to reference the tables.

1. Member tables (or element tables)

A member table is a complete list of the elements of a dimension.

Table can have any name. Stored in _Dimensions.Param1.

Column	Type	Description
MemberName	String(71)	The name of the element.
MemberType	String(1)	The type of the element. • N = Numeric • S = String • C = Consolidated
MemberOrder	Integer	The numeric position of the element within the dimension.
Attribute Columns	Any	Optional. Describe the attributes of the dimension. • There can be any number of these up to 50; there can be none. • These columns can be distributed anywhere and need not appear together. • They must be listed in the _AttributeFields table in the rows belonging to their dimension. • Their attribute tables (which are OLAP Server tables index by a number 1, 2 or 3 and not relational tables) are also specified in the _AttributeFields table.
Columns for Relations/Hierarchy	As in Relationships Table.	Optional. • Use if the dimension has a hierarchy and if a separate Relationships Table is not used. • If this table is used for the relations, then the name of this table must be entered in the column Param2 of _Dimensions.

Comments

- The dimension is loaded with the data in this table as rows.
- The table can have any name, but this name must be stored in the _Dimensions table in the column Param1 in the row of the dimension whose name is found in the column DimensionName.
- Attribute names must be stored in the _AttributeFields table for the dimension so the OLAP Server knows to which dimension the attribute data relates.
- Columns can be optionally used to describe the dimension's element hierarchy. If used, they must be identical to the columns of the relations table and would be used as an alternative to using a separate relations table. The advantage of placing the relations here (and avoiding a separate table) is system simplicity. The disadvantage is that this does not allow multiple parents of an element.

The name of this member table must appear in _Dimensions.Param2 (as well as in _Dimensions.Param1).

- A dimension is loaded with its elements from its member table using JobType 1 in _Jobs with the dimension name in _Jobs.JobObject. So are any relationships in this table loaded provided the name of this table is in _Dimensions.Param2.
- Any attributes are also loaded with _Jobs.JobType 1. Attributes can be loaded separately with _Jobs.JobType = 6. In both these cases the dimension's name must be in _Jobs.JobObject.
- A dimension can have no more than 50 attributes. Therefore, there should be no more than 50 attribute columns in the member table.

Member Table Creation with SQL Server

```
CREATE TABLE [dbo].[my_member_table /* Generic name */](
 [MemberName] [nvarchar](71) NOT NULL,
 [MemberType] [nvarchar](1) NOT NULL,
 [MemberOrder] [int] NULL
 /* Attribute Columns. Can be anywhere. */
 /* Columns from relations table if this does not exist. Can be
anywhere. */
)
```

Member Table Creation with Oracle

```
CREATE TABLE "my_member_table" /* Generic name */(
 "MemberName" nvarchar2(71) NOT NULL,
 "MemberType" nvarchar2(1) NOT NULL,
 "MemberOrder" int NULL
 /* Attribute Columns. Can be anywhere. */
 /* Columns from relations table if this does not exist. Can be
anywhere. */
);
```

2. Relationship tables (or hierarchy tables)

A Relationship table is a list of the parent-child relationships between the elements of a dimension.

Table can have any name. Stored in _Dimensions.Param2.

Column	Type	Description
MemberName	String(71)	Name of child element. Must be present in member table.
ParentMemberName	String(71)	Name of parent element. Must be present in member table.

Column	Type	Description
Weight	Real	The weighting given to data on this element within this parent-child relationship. If NULL, take the value to be 1.
MemberParentOrder	Integer	The numeric position of the child element among the parent's children. If NULL, take the value to be 0.

Comments

- This table is an alternative to using its corresponding member table to store relationships. But if this is used, it is more flexible. A child can have any number of parents. (A parent can have any number of children in any case; this also holds if the member table is used.)
- The name of this table must be in _Dimensions.Param2 in the row for the dimension whose name is found in DimensionName.
- A dimension is loaded with its relationships using JobType 1 in _Jobs with the dimension name in _Jobs.JobObject. The relationships are loaded automatically with the elements where these relationships exist and if the name of this table is in _Dimensions.Param2.

Relationship Table Creation with SQL Server

```
CREATE TABLE [dbo].[my_relationship_table  /* Generic name */](
   [MemberName] [nvarchar](71) NOT NULL,
   [ParentMemberName] [nvarchar](71) NULL,
   [Weight] [real] NULL,
   [MemberParentOrder] [int] NULL
)
```

Relationship Table Creation with Oracle

```
CREATE TABLE "my_relationship_table"  /* Generic name */(
   "MemberName" nvarchar2(71) NOT NULL,
   "ParentMemberName" nvarchar2(71) NULL,
   "Weight" binary_double NULL,
   "MemberParentOrder" int NULL
);
```

3. Subset tables

A subset table is an ordered list of the elements of a dimension, a subset of the complete set.

Table can have any name. Stored in _Subsets.SubsetTable.

Column	Type	Description
MemberName	String(71)	The name of the element.

Column	Type	Description
MemberOrder	Integer	The numeric position of the element within the dimension.

Comments

- A subset table is the list of elements of one kind of dimension subset only: a static element list.
- It has the same structure as a member table, but without possible attributes or relationships.
- A subset table's name must be stored in the column SubsetTable of _Subsets in the same row as its dimension's name, which is stored in DimName, and its Olap name stored in SubsetName.
- Subsets of a dimension are loaded automatically with the loading of their dimension. This would be achieved with _Jobs.JobType = 1 and the dimension name in _Jobs.JobObject.
- Subsets of a dimension can be loaded separately from the loading of their dimension with _Jobs.JobType = 8 and the dimension name in _Jobs.JobObject.

Subset Table Creation with SQL Server

```
CREATE TABLE [dbo].[my_subset_table /* Generic name */](
   [MemberName] [nvarchar](71) NOT NULL,
   [MemberOrder] [int] NULL
)
```

Subset Table Creation with Oracle

```
CREATE TABLE "my_subset_table" /* Generic name */(
   "MemberName" nvarchar2(71) NOT NULL,
   "MemberOrder" int NULL
)
```

4. Fact tables

A fact table contains the facts to load and relates them to their coordinates defined by dimension elements.

Name stored in _FactLoad.FactTable.

Column	Type	Description
Names of Columns matching Cube's Dimensions	String(71)	Element names of the columns' matching dimensions. • There could be any number of these columns (there could even be none) • See comments

Column	Type	Description
Name of Column describing Cube's Facts	Real or String(MAX)	Fact at the coordinates in the cube defined by the elements in the columns and any possible target elements defined in _FactLoadParameters. The name of this column is defined in the RelationColumnName column of _FactLoads. **Note:** • Multiple rows of numeric facts with identical element coordinates are aggregated. • Last row only accepted if there are multiple rows of string facts with identical element coordinates. • A NULL value is taken as 0 for numeric and "" for character strings.
DataVersion	BigInteger	Version number of the data. For incremental load only (_FactLoad.FactLoadType = 3). For other load types this must be NULL if the column DataVersion exists (which it need not). See Comments and "_FactLoad" on page 202 table for more information in this.

Comments

- A fact load needs two objects to define it:
  - A target cube
  - A source table.

  These are identified by a single job object. The name of the job object is found in JobObject of _FactLoad, the name the cube in CubeId and the name of the table (described above) in FactTable.
- The elements of the dimensions defining the coordinate positions at which to load a fact appear in two possible places:
  - The columns of the fact table matching the elements' dimensions
  - As target elements in _FactLoadParameters, where their dimensions are each expressed as a zero based index in _FactLoadParameters.DimIdx and each target element itself in _FactLoadParameters.TargetElement for its dimension.

- Elements could be qualified with the appropriate hierarchy name, with the two separated with a tab character.
  - A hierarchy-qualified element would be written in the fact table as:

    hierarchyName<tab>elementName

- All elements but target elements can possibly appear in the fact table. (A 2-dimensional cube with 2 target elements needs just one column, whose name is _FactLoads.RelationalColumnName, and a single row to load it.)
- The whole table might correspond to a single element of one or more dimensions. These would be target elements in _FactLoadParameters. The columns in the fact table would not exist for such elements.
- A loading of facts from a fact table takes place with _Jobs.JobType = 3. _Jobs.JobsObject is the job object identifying the load from table to cube.
- For an incremental load the row is only selected if _FactLoad.FactChangesTable is NULL (meaning that there is no fact changes table) and DataVersion falls in the range _FactLoad.LastLoadedVersion < DataVersion <= _FactLoad.NewestVersionToLoad. The row is then linked to all other rows with the same element data and the total computed. This total is written to the cube to the cells whose coordinated match the element combination of the linked rows.
- Tip for Loading Cubes Containing both Numeric and String Facts

  For loading cubes containing both numeric and string data, separate load processes will be needed for the numerics and strings. This is because only one fact column is allowed in a fact table, which can only be either numeric or string. Then, separate fact load data in _FactLoads will need to be set up as partial loads. And the job parameters of the measure dimension corresponding to the numeric and string elements appropriately in separate parameter sets. See the table _JobsParameters for details on how this is done.

Fact Table Creation with SQL Server

For numeric data:

```
CREATE TABLE [dbo].[my_fact_table /* Generic name */](
   [my_column_1] [nvarchar](71) NOT NULL,  /* Any number of these */
   [my_column_2] [nvarchar](71) NOT NULL,
   [my_column_n] [nvarchar](71) NOT NULL,
   [my_fact_column] [real] NULL, /* Must be present */
   [DataVersion] [bigint] NULL /* Optional */
)
```

OR, for string data:

```
CREATE TABLE [dbo].[my_fact_table /* Generic name */](
   [my_column_1] [nvarchar](71) NOT NULL,  /* Any number of these */
   [my_column_2] [nvarchar](71) NOT NULL,
   [my_column_n] [nvarchar](71) NOT NULL,
   [my_fact_column] [nvarchar](MAX) NULL, /* Must be present */
   [DataVersion] [bigint] NULL /* Optional */
)
```

Fact Table Creation with Oracle

For numeric data:

```
CREATE TABLE “my_fact_table” /* Generic name */(
 “my_column_1” nvarchar2(71) NOT NULL,  /* Any number of these */
 “my_column_2” nvarchar2(71) NOT NULL,
 “my_column_n” nvarchar2(71) NOT NULL,
 “my_fact_column” binary_double NULL, /* Must be present */
 “DataVersion” number(19, 0) NULL /* Optional */
);
```

OR, for string data:

```
CREATE TABLE “my_fact_table” /* Generic name */(
 “my_column_1” nvarchar2(71) NOT NULL,  /* Any number of these */
 “my_column_2” nvarchar2(71) NOT NULL,
 “my_column_n” nvarchar2(71) NOT NULL,
 “my_fact_column” nvarchar2(2000) NULL, /* Must be present */
 “DataVersion” number(19, 0) NULL /* Optional */
);
```

5. Fact changes tables

Fact changes tables are only used for incremental loads for which _FactLoad.FactLoadType = 3 and _FactLoad.FactChangesTable != NULL. The name of this table is stored in the field _FactLoad.FactLoadChanges.

Name stored in _FactLoad.FactChangesTable.

Column	Type	Description
Names of Columns matching Cube's Dimensions	String(71)	Element names of the columns' matching dimensions. • There could be any number of these columns (there could even be none) • Pattern same as for Fact Table. See "4. Fact tables" on page 224.
DataVersion	BigInteger	Version number of the data

- DataVersion is selected according to the criteria in the _FactLoad table. The row will be selected if _FactLoad.LastLoadedVersion < DataVersion <= NewestVersionToLoad.
- The element data in the columns is then used as a key to select those rows in the fact table to load. Numeric values are aggregated and the total is written to the cube to the cell whose coordinates matches the element combination selected.

Related error messages

Error code	Meaning
294	Relational DB error
295	Structural changes are only possible in single-node cluster mode or if cluster mode is dsabled.
296	Invalid cube configuration in table _Cubes
297	Invalid cube dimension data in table _CubesDimensions
298	Invalid cube access control data in table _CubeAccessControl
299	Invalid dimension data in table _Dimensions
300	Invalid attribute field data in table _AttributeFields
301	Invalid subset data in table _Subsets
302	Invalid DBLoad source connection data
303	Invalid cube metadata
304	Invalid job data
305	General SQL error in DBLoad process
306	General XML error in DBLoad process
307	Invalid fact load data
308	There are too many elements in the diemsion to use cache partitioning.
309	Invalid DAC or MDAC data
310	Invalid DBLoad source accessor setup
311	Invalid Cron schedule string
312	Element attribute is too big to load into Olap

Data type translation table

The generic data types used in this manual are not specific to any database type. The following table translates them to SQL Server. In later versions of the server, translations to Oracle will also be given and Oracle. Note that the SQL Server types `nvarchar` and `ntext` are preferred to `varchar` and `text`, and the Oracle types `nvarchar2` and `nclob` to `varchar2` and `clob` because they allow the use of non-Ascii characters.

Note: In SQL Server `nvarchar` and `ntext` are preferred to `varchar` and text because they allow the use of non-Ascii characters.

Generic Type	SQL Server Type	Oracle Type
Integer	int	number
BigInteger	bigint	number(19, 0)
String(n)	nvarchar(n) (limited to 2000)	nvarchar2(n) (limited to 2000)
	nvarchar(MAX)	nclob
Real	real	binary_double
DateTime	datetime	Timestamp
Boolean	bit	Number
Text	ntext	Nclob

SQL Server script

Below is the SQL script that can be used:

- For creating the metadata tables by using the metadata part of the script directly.
- As a guide for creating the data tables, bearing in mind related metadata settings.

In addition the files `CreateOlapMetadata.sql` and `CreateOlapMetadataOracle.sql` containing the sample code for SQL Server and Oracle are installed with the OLAP Server Setup in the `bin64` folder of OLAP Server (default path: `C:\Program Files\Infor\BI\OLAP\bin64`).

```
/*
**          SQL Server
**
**  Version 11.0.0
**
**  Create the basic OLAP metadata tables:
**    _AttributeFields
**    _CubeAccessControl
**    _Cubes
**    _CubesDimensions
**    _Dimensions
**    _DimensionLevelNames
**    _Jobs
**    _JobsParameters
**    _FactLoad
**    _FactLoadParameters
**    _Hierarchies
**    _Parameters
**    _ScheduledJobs
**    _Sources
**    _Subsets
**    _ErrorText
```

```
**    _ErrorLogFactLoad
**    _ErrorLogElementLoad
**    _ErrorLogAttributeFill
**    _ErrorLogOther
*/

CREATE TABLE [dbo].[_AttributeFields](
    [DimensionName] [nvarchar](50) NOT NULL,
    [TableId] [int] NOT NULL,
    [RelationalField] [nvarchar](50) NOT NULL,
    [FieldName] [nvarchar](50) NOT NULL,
    [FieldDescription] [nvarchar](150) NULL,
    [FieldType] [nchar](1) NULL,
    [FieldWidth] [int] NOT NULL,
    [FieldDecimalPlaces] [int] NULL,
    [FieldOrderPosition] [int] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_CubeAccessControl](
    [CubeName] [nvarchar](50) NOT NULL,
    [AccessCubeName] [nvarchar](50) NOT NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_Cubes](
    [CubeName] [nvarchar](50) NOT NULL,
    [CubeDescription] [nvarchar](150) NOT NULL,
    [Param1] [nvarchar](150) NULL,
    [Param2] [nvarchar](250) NULL,
    [AccessControl] [bit] NOT NULL,
    [TransactionLog] [bit] NOT NULL,
    [CubeType] [nchar](1) NULL,
    [ExtendedProps] [ntext] NULL,
    [CubeRules] [ntext] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_CubesDimensions](
    [CubeName] [nvarchar](50) NOT NULL,
    [DimensionName] [nvarchar](50) NOT NULL,
    [DimensionOrder] [int] NOT NULL,
    [MeasureDimension] [bit] NOT NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_Dimensions](
    [DimensionName] [nvarchar](50) NOT NULL,
    [DimensionDescription] [nvarchar](150) NOT NULL,
    [SourceId] [int] NULL,
    [Param1] [ntext] NULL,
    [Param2] [ntext]NULL,
    [DimensionType] [int] NULL,
```

```
    [DefaultElement] [nvarchar](71) NULL,
    [FlatView] [bit] NOT NULL,
    [InvertedHierarchy] [bit] NOT NULL,
    [AccessCube] [nvarchar](50) NULL,
    [LoadFlags] [bigint] NULL,
    [ExtendedProps] [ntext] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_DimensionLevelNames](
    [DimensionName] [nvarchar](50) NOT NULL,
    [LevelNumber] [int] NOT NULL,
    [LevelName] [nvarchar](50) NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_Jobs](
    [JobId] [bigint] NOT NULL UNIQUE,
    [JobType] [int] NOT NULL,
    [JobObject] [nvarchar](71) NOT NULL,
    [JobParameter] [int] NULL,
    [JobGroup] [nvarchar](100) NULL,
    [Started] [datetime] NULL,
    [Completed] [datetime] NULL,
    [Status] [int] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_JobsParameters](
    [JobParameter] [bigint] NOT NULL,
    [ParameterSet] [int] NOT NULL,
    [DimIdx] [int] NOT NULL,
    [ParameterValue] [ntext] NOT NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_FactLoad](
    [JobObject] [nvarchar](50) NOT NULL UNIQUE,
    [CubeId] [nvarchar](50) NOT NULL,
    [FactTable] [ntext] NULL,
    [FactChangesTable] [ntext] NULL,
    [FactLoadType] [int] NOT NULL,
    [ParameterId] [int] NOT NULL,
    [LastLoadedVersion] [int] NULL,
    [NewestVersionToLoad] [int] NULL,
    [SourceId] [int] NULL,
    [FactType] [int] NULL,
    [RelationalColumnName] [nvarchar](50) NULL,
    [LoadFlags] [bigint] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_FactLoadParameters](
```

```
    [Id] [int] NOT NULL,
    [DimIdx] [int] NOT NULL,
    [ColumnName] [nvarchar](50) NULL,
    [TargetElement] [nvarchar](71) NULL,
    [TargetElementUnknown] [nvarchar](71) NULL,
    [TargetElementMissing] [nvarchar](71) NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_Hierarchies](
    [HierarchyName] [nvarchar](50) NOT NULL,
    [HierarchyDescription] [nvarchar](150) NOT NULL,
    [DimensionName] [nvarchar](50) NOT NULL,
    [SourceId] [int] NULL,
    [HierarchyType] [int] NOT NULL,
    [Param1] [ntext] NULL,
    [Param2] [ntext] NULL,
    [DefaultElement] [nvarchar](71) NULL,
    [ExtendedProps] [ntext] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_Parameters](
    [Id] [bigint] NOT NULL,
    [Key] [nvarchar](50) NOT NULL,
    [DataType] [int] NOT NULL,
    [StringValue] [ntext] NULL,
    [IntegerValue] [int] NULL,
    [LinkId] [bigint] NULL,
    [Comment] [ntext] NULL
    CONSTRAINT ID_KEY_PK PRIMARY KEY ([Id], [Key])
)

CREATE TABLE [dbo].[_ScheduledJobs](
    [Id] [bigint] NOT NULL UNIQUE,
    [JobType] [int] NOT NULL,
    [JobObject] [nvarchar](71) NOT NULL,
    [JobParameter] [int] NULL,
    [Schedule] [ntext] NOT NULL,
    [Scheduled] [bit] NOT NULL,
    [Disabled] [bit] NOT NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_Sources](
    [SourceId] [int] NOT NULL,
    [SourceType] [int] NOT NULL,
    [ConnectionParameters] [ntext] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_Subsets](
    [DimName] [nvarchar](50) NOT NULL,
```

```
    [SubsetName] [nvarchar](50) NOT NULL,
    [SubsetKind] [int] NOT NULL,
    [SubsetCategory] [int] NOT NULL,
    [SubsetTable] [ntext] NOT NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_ErrorText](
    [ErrorCode] [int] NOT NULL,
    [LanguageRegion] [nvarchar](15) NOT NULL,
    [Text] [ntext] NULL
)

CREATE TABLE [dbo].[_ErrorLogFactLoad](
    [JobId] [bigint] NOT NULL,
    [ErrorCode] [int] NOT NULL,
    [ErrorDimIdx] [int] NULL,
    [ErrorDimName] [nvarchar](50) NULL,
    [Element00] [nvarchar](256) NULL,
    [Element01] [nvarchar](256) NULL,
    [Element02] [nvarchar](256) NULL,
    [Element03] [nvarchar](256) NULL,
    [Element04] [nvarchar](256) NULL,
    [Element05] [nvarchar](256) NULL,
    [Element06] [nvarchar](256) NULL,
    [Element07] [nvarchar](256) NULL,
    [Element08] [nvarchar](256) NULL,
    [Element09] [nvarchar](256) NULL,
    [Element10] [nvarchar](256) NULL,
    [Element11] [nvarchar](256) NULL,
    [Element12] [nvarchar](256) NULL,
    [Element13] [nvarchar](256) NULL,
    [Element14] [nvarchar](256) NULL,
    [Element15] [nvarchar](256) NULL,
    [Element16] [nvarchar](256) NULL,
    [Element17] [nvarchar](256) NULL,
    [Element18] [nvarchar](256) NULL,
    [Element19] [nvarchar](256) NULL,
    [Element20] [nvarchar](256) NULL,
    [Element21] [nvarchar](256) NULL,
    [Element22] [nvarchar](256) NULL,
    [Element23] [nvarchar](256) NULL,
    [Element24] [nvarchar](256) NULL,
    [Element25] [nvarchar](256) NULL,
    [Element26] [nvarchar](256) NULL,
    [Element27] [nvarchar](256) NULL,
    [Element28] [nvarchar](256) NULL,
    [Element29] [nvarchar](256) NULL,
    [ElementCaption00] [nvarchar](256) NULL,
    [ElementCaption01] [nvarchar](256) NULL,
    [ElementCaption02] [nvarchar](256) NULL,
    [ElementCaption03] [nvarchar](256) NULL,
    [ElementCaption04] [nvarchar](256) NULL,
```

```
    [ElementCaption05] [nvarchar](256) NULL,
    [ElementCaption06] [nvarchar](256) NULL,
    [ElementCaption07] [nvarchar](256) NULL,
    [ElementCaption08] [nvarchar](256) NULL,
    [ElementCaption09] [nvarchar](256) NULL,
    [ElementCaption10] [nvarchar](256) NULL,
    [ElementCaption11] [nvarchar](256) NULL,
    [ElementCaption12] [nvarchar](256) NULL,
    [ElementCaption13] [nvarchar](256) NULL,
    [ElementCaption14] [nvarchar](256) NULL,
    [ElementCaption15] [nvarchar](256) NULL,
    [ElementCaption16] [nvarchar](256) NULL,
    [ElementCaption17] [nvarchar](256) NULL,
    [ElementCaption18] [nvarchar](256) NULL,
    [ElementCaption19] [nvarchar](256) NULL,
    [ElementCaption20] [nvarchar](256) NULL,
    [ElementCaption21] [nvarchar](256) NULL,
    [ElementCaption22] [nvarchar](256) NULL,
    [ElementCaption23] [nvarchar](256) NULL,
    [ElementCaption24] [nvarchar](256) NULL,
    [ElementCaption25] [nvarchar](256) NULL,
    [ElementCaption26] [nvarchar](256) NULL,
    [ElementCaption27] [nvarchar](256) NULL,
    [ElementCaption28] [nvarchar](256) NULL,
    [ElementCaption29] [nvarchar](256) NULL,
    [FactType] [int] NULL,
    [ValueNumeric] [real] NULL,
    [ValueString] [ntext] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_ErrorLogElementLoad](
    [JobId] [bigint] NOT NULL,
    [ErrorCode] [int] NOT NULL,
    [Dimension] [nvarchar](256) NULL,
    [DimensionCaption] [nvarchar](256) NULL,
    [Hierarchy] [nvarchar](256) NULL,
    [HierarchyCaption] [nvarchar](256) NULL,
    [Element] [nvarchar](256) NULL,
    [ElementCaption] [nvarchar](256) NULL,
    [ElementType] [nvarchar](256) NULL,
    [ParentElement] [nvarchar](256) NULL,
    [ParentElementCaption] [nvarchar](256) NULL,
    [Weight] [real] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_ErrorLogAttributeFill] (
    [JobId] [bigint] NOT NULL,
    [ErrorCode] [int] NOT NULL,
    [Dimension] [nvarchar](256) NULL,
    [DimensionCaption] [nvarchar](256) NULL,
    [Hierarchy] [nvarchar](256) NULL,
```

```
    [HierarchyCaption] [nvarchar](256) NULL,
    [Element] [nvarchar](256) NULL,
    [ElementCaption] [nvarchar](256) NULL,
    [TableId] [int] NULL,
    [AttributeFieldName] [nvarchar](256) NULL,
    [AttributeDataValue] [nvarchar](512) NULL,
    [AttributeFieldType] [nvarchar](16) NULL,
    [AttributeFieldLength] [int] NULL,
    [AttributeFieldDecimals] [int] NULL,
    [Comment] [ntext] NULL
)

CREATE TABLE [dbo].[_ErrorLogOther](
    [JobId] [bigint] NOT NULL,
    [ErrorCode] [int] NOT NULL,
    [ObjectKind] [nvarchar](256) NULL,
    [ObjectName] [nvarchar](256) NULL,
    [ObjectCaption] [nvarchar](256) NULL,
    [ErrorText] [text] NULL
)
```

Oracle script

Below is the Oracle script that can be used for creating the metadata tables directly.

```
/*
**          Oracle
**
**  Version 11.0.0
**
**  Create the basic OLAP metadata tables:
**    _AttributeFields
**    _CubeAccessControl
**    _Cubes
**    _CubesDimensions
**    _Dimensions
**    _DimensionLevelNames
**    _Jobs
**    _JobsParameters
**    _FactLoad
**    _FactLoadParameters
**    _Hierarchies
**    _Parameters
**    _ScheduledJobs
**    _Sources
**    _Subsets
**    _ErrorText
**    _ErrorLogFactLoad
```

```
**    _ErrorLogElementLoad
**    _ErrorLogAttributeFill
**    _ErrorLogOther
*/

CREATE TABLE "_AttributeFields"(
    "DimensionName" nvarchar2(50) NOT NULL,
    "TableId" int NOT NULL,
    "RelationalField" nvarchar2(50) NOT NULL,
    "FieldName" nvarchar2(50) NOT NULL,
    "FieldDescription" nvarchar2(150) NULL,
    "FieldType" nchar(1) NULL,
    "FieldWidth" int NOT NULL,
    "FieldDecimalPlaces" int NULL,
    "FieldOrderPosition" int NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_CubeAccessControl"(
    "CubeName" nvarchar2(50) NOT NULL,
    "AccessCubeName" nvarchar2(50) NOT NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_Cubes"(
    "CubeName" nvarchar2(50) NOT NULL,
    "CubeDescription" nvarchar2(150) NOT NULL,
    "Param1" nvarchar2(150) NULL,
    "Param2" nvarchar2(250) NULL,
    "AccessControl" number NOT NULL,
    "TransactionLog" number NOT NULL,
    "CubeType" nchar(1) NULL,
    "ExtendedProps" nclob NULL,
    "CubeRules" nclob NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_CubesDimensions"(
    "CubeName" nvarchar2(50) NOT NULL,
    "DimensionName" nvarchar2(50) NOT NULL,
    "DimensionOrder" int NOT NULL,
    "MeasureDimension" number NOT NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_Dimensions"(
    "DimensionName" nvarchar2(50) NOT NULL,
    "DimensionDescription" nvarchar2(150) NOT NULL,
    "SourceId" int NULL,
    "Param1" nclob NULL,
    "Param2" nclob NULL,
    "DimensionType" int NULL,
    "DefaultElement" nvarchar2(71) NULL,
```

```
    "FlatView" number NOT NULL,
    "InvertedHierarchy" number NOT NULL,
    "AccessCube" nvarchar2(50) NULL,
    "LoadFlags" number(19, 0) NULL,
    "ExtendedProps" nclob NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_DimensionLevelNames"(
    "DimensionName" nvarchar2(50) NOT NULL,
    "LevelNumber" int NULL,
    "LevelName" nvarchar2(50) NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_Jobs"(
    "JobId" number(19, 0) NOT NULL UNIQUE,
    "JobType" int NOT NULL,
    "JobObject" nvarchar2(71) NOT NULL,
    "JobParameter" int NULL,
    "JobGroup" nvarchar2(100) NULL,
    "Started" timestamp(6) NULL,
    "Completed" timestamp(6) NULL,
    "Status" int NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_JobsParameters"(
    "JobParameter" number(19, 0) NOT NULL,
    "ParameterSet" int NOT NULL,
    "DimIdx" int NOT NULL,
    "ParameterValue" nclob NOT NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_FactLoad"(
    "JobObject" nvarchar2(50) NOT NULL UNIQUE,
    "CubeId" nvarchar2(50) NOT NULL,
    "FactTable" nclob NULL,
    "FactChangesTable" nclob NULL,
    "FactLoadType" int NOT NULL,
    "ParameterId" int NOT NULL,
    "LastLoadedVersion" int NULL,
    "NewestVersionToLoad" int NULL,
    "SourceId" int NULL,
    "FactType" int NULL,
    "RelationalColumnName" nvarchar2(50) NULL,
    "LoadFlags" number(19, 0) NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_FactLoadParameters"(
    "Id" int NOT NULL,
```

```
    "DimIdx" int NOT NULL,
    "ColumnName" nvarchar2(50) NULL,
    "TargetElement" nvarchar2(71) NULL,
    "TargetElementUnknown" nvarchar2(71) NULL,
    "TargetElementMissing" nvarchar2(71) NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_Hierarchies"(
    "HierarchyName" nvarchar2(50) NOT NULL,
    "HierarchyDescription" nvarchar2(150) NOT NULL,
    "DimensionName" nvarchar2(50) NOT NULL,
    "SourceId" int NULL,
    "HierarchyType" int NOT NULL,
    "Param1" nclob NULL,
    "Param2" nclob NULL,
    "DefaultElement" nvarchar2(71) NULL,
    "ExtendedProps" nclob NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_Parameters"(
    "Id" number(38, 0) NOT NULL,
    "Key" nvarchar2(50) NOT NULL,
    "DataType" int NOT NULL,
    "StringValue" nclob NULL,
    "IntegerValue" int NULL,
    "LinkId" number(38, 0) NULL,
    "Comment" nclob NULL,
    CONSTRAINT ID_KEY_PK PRIMARY KEY ("Id", "Key")
);

CREATE TABLE "_ScheduledJobs"(
    "Id" number(19, 0) NOT NULL UNIQUE,
    "JobType" int NOT NULL,
    "JobObject" nvarchar2(71) NOT NULL,
    "JobParameter" int NULL,
    "Schedule" nclob NOT NULL,
    "Scheduled" number NOT NULL,
    "Disabled" number NOT NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_Sources"(
    "SourceId" int NOT NULL,
    "SourceType" int NOT NULL,
    "ConnectionParameters" nclob NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_Subsets"(
    "DimName" nvarchar2(50) NOT NULL,
    "SubsetName" nvarchar2(50) NOT NULL,
```

```
    "SubsetKind" int NOT NULL,
    "SubsetCategory" int NOT NULL,
    "SubsetTable" nclob NOT NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_ErrorText"(
    "ErrorCode" int NOT NULL,
    "LanguageRegion" nvarchar2(15) NOT NULL,
    "Text" nclob NULL
);

CREATE TABLE "_ErrorLogFactLoad"(
    "JobId" number(19, 0) NOT NULL,
    "ErrorCode" int NOT NULL,
    "ErrorDimIdx" int NULL,
    "ErrorDimName" nvarchar2(50) NULL,
    "Element00" nvarchar2(256) NULL,
    "Element01" nvarchar2(256) NULL,
    "Element02" nvarchar2(256) NULL,
    "Element03" nvarchar2(256) NULL,
    "Element04" nvarchar2(256) NULL,
    "Element05" nvarchar2(256) NULL,
    "Element06" nvarchar2(256) NULL,
    "Element07" nvarchar2(256) NULL,
    "Element08" nvarchar2(256) NULL,
    "Element09" nvarchar2(256) NULL,
    "Element10" nvarchar2(256) NULL,
    "Element11" nvarchar2(256) NULL,
    "Element12" nvarchar2(256) NULL,
    "Element13" nvarchar2(256) NULL,
    "Element14" nvarchar2(256) NULL,
    "Element15" nvarchar2(256) NULL,
    "Element16" nvarchar2(256) NULL,
    "Element17" nvarchar2(256) NULL,
    "Element18" nvarchar2(256) NULL,
    "Element19" nvarchar2(256) NULL,
    "Element20" nvarchar2(256) NULL,
    "Element21" nvarchar2(256) NULL,
    "Element22" nvarchar2(256) NULL,
    "Element23" nvarchar2(256) NULL,
    "Element24" nvarchar2(256) NULL,
    "Element25" nvarchar2(256) NULL,
    "Element26" nvarchar2(256) NULL,
    "Element27" nvarchar2(256) NULL,
    "Element28" nvarchar2(256) NULL,
    "Element29" nvarchar2(256) NULL,
    "ElementCaption00" nvarchar2(256) NULL,
    "ElementCaption01" nvarchar2(256) NULL,
    "ElementCaption02" nvarchar2(256) NULL,
    "ElementCaption03" nvarchar2(256) NULL,
    "ElementCaption04" nvarchar2(256) NULL,
    "ElementCaption05" nvarchar2(256) NULL,
```

```
    "ElementCaption06" nvarchar2(256) NULL,
    "ElementCaption07" nvarchar2(256) NULL,
    "ElementCaption08" nvarchar2(256) NULL,
    "ElementCaption09" nvarchar2(256) NULL,
    "ElementCaption10" nvarchar2(256) NULL,
    "ElementCaption11" nvarchar2(256) NULL,
    "ElementCaption12" nvarchar2(256) NULL,
    "ElementCaption13" nvarchar2(256) NULL,
    "ElementCaption14" nvarchar2(256) NULL,
    "ElementCaption15" nvarchar2(256) NULL,
    "ElementCaption16" nvarchar2(256) NULL,
    "ElementCaption17" nvarchar2(256) NULL,
    "ElementCaption18" nvarchar2(256) NULL,
    "ElementCaption19" nvarchar2(256) NULL,
    "ElementCaption20" nvarchar2(256) NULL,
    "ElementCaption21" nvarchar2(256) NULL,
    "ElementCaption22" nvarchar2(256) NULL,
    "ElementCaption23" nvarchar2(256) NULL,
    "ElementCaption24" nvarchar2(256) NULL,
    "ElementCaption25" nvarchar2(256) NULL,
    "ElementCaption26" nvarchar2(256) NULL,
    "ElementCaption27" nvarchar2(256) NULL,
    "ElementCaption28" nvarchar2(256) NULL,
    "ElementCaption29" nvarchar2(256) NULL,
    "FactType" int NULL,
    "ValueNumeric" real NULL,
    "ValueString" nclob NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_ErrorLogElementLoad"(
    "JobId" number(19, 0) NOT NULL,
    "ErrorCode" int NOT NULL,
    "Dimension" nvarchar2(256) NULL,
    "DimensionCaption" nvarchar2(256) NULL,
    "Hierarchy" nvarchar2(256) NULL,
    "HierarchyCaption" nvarchar2(256) NULL,
    "Element" nvarchar2(256) NULL,
    "ElementCaption" nvarchar2(256) NULL,
    "ElementType" nvarchar2(256) NULL,
    "ParentElement" nvarchar2(256) NULL,
    "ParentElementCaption" nvarchar2(256) NULL,
    "Weight" real NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_ErrorLogAttributeFill" (
    "JobId" number(19, 0) NOT NULL,
    "ErrorCode" int NOT NULL,
    "Dimension" nvarchar2(256) NULL,
    "DimensionCaption" nvarchar2(256) NULL,
    "Hierarchy" nvarchar2(256) NULL,
    "HierarchyCaption" nvarchar2(256) NULL,
```

```
    "Element" nvarchar2(256) NULL,
    "ElementCaption" nvarchar2(256) NULL,
    "TableId" int NULL,
    "AttributeFieldName" nvarchar2(256) NULL,
    "AttributeDataValue" nvarchar2(512) NULL,
    "AttributeFieldType" nvarchar2(16) NULL,
    "AttributeFieldLength" int NULL,
    "AttributeFieldDecimals" int NULL,
    "Comment" nclob NULL
);

CREATE TABLE "_ErrorLogOther"(
    "JobId" number(19, 0) NOT NULL,
    "ErrorCode" int NOT NULL,
    "ObjectKind" nvarchar2(256) NULL,
    "ObjectName" nvarchar2(256) NULL,
    "ObjectCaption" nvarchar2(256) NULL,
    "ErrorText" nclob NULL
);
```

Postgres Script

Below is the Postgres script that can be used for creating the metadata tables directly.

```
/*
**          Postgres
**
**  Version 11.0.0
**
**  Create the basic OLAP metadata tables:
**    _AttributeFields
**    _CubeAccessControl
**    _Cubes
**    _CubesDimensions
**    _Dimensions
**    _DimensionLevelNames
**    _Jobs
**    _JobsParameters
**    _FactLoad
**    _FactLoadParameters
**    _Hierarchies
**    _Parameters
**    _ScheduledJobs
**    _Sources
**    _Subsets
**    _ErrorText
**    _ErrorLogFactLoad
**    _ErrorLogElementLoad
```

```
**    _ErrorLogAttributeFill
**    _ErrorLogOther
*/

CREATE TABLE "_AttributeFields"(
    "DimensionName" varchar(50) NOT NULL,
    "TableId" int NOT NULL,
    "RelationalField" varchar(50) NOT NULL,
    "FieldName" varchar(50) NOT NULL,
    "FieldDescription" varchar(150) NULL,
    "FieldType" char(1) NULL,
    "FieldWidth" int NOT NULL,
    "FieldDecimalPlaces" int NULL,
    "FieldOrderPosition" int NULL,
    "Comment" text NULL
);

CREATE TABLE "_CubeAccessControl"(
    "CubeName" varchar(50) NOT NULL,
    "AccessCubeName" varchar(50) NOT NULL,
    "Comment" text NULL
);

CREATE TABLE "_Cubes"(
    "CubeName" varchar(50) NOT NULL,
    "CubeDescription" varchar(150) NOT NULL,
    "Param1" varchar(150) NULL,
    "Param2" varchar(250) NULL,
    "AccessControl" int NOT NULL,
    "TransactionLog" int NOT NULL,
    "CubeType" char(1) NULL,
    "ExtendedProps" text NULL,
    "CubeRules" text NULL,
    "Comment" text NULL
);

CREATE TABLE "_CubesDimensions"(
    "CubeName" varchar(50) NOT NULL,
    "DimensionName" varchar(50) NOT NULL,
    "DimensionOrder" int NOT NULL,
    "MeasureDimension" int NOT NULL,
    "Comment" text NULL
);

CREATE TABLE "_Dimensions"(
    "DimensionName" varchar(50) NOT NULL,
    "DimensionDescription" varchar(150) NOT NULL,
    "SourceId" int NULL,
    "Param1" text NULL,
    "Param2" text NULL,
    "DimensionType" int NULL,
    "DefaultElement" varchar(71) NULL,
    "FlatView" int NOT NULL,
```

```
    "InvertedHierarchy" int NOT NULL,
    "AccessCube" varchar(50) NULL,
    "LoadFlags" bigint NULL,
    "ExtendedProps" text NULL,
    "Comment" text NULL
);

CREATE TABLE "_DimensionLevelNames"(
    "DimensionName" varchar(50) NOT NULL,
    "LevelNumber" int NULL,
    "LevelName" varchar(50) NULL,
    "Comment" text NULL
);

CREATE TABLE "_Jobs"(
    "JobId" bigint NOT NULL UNIQUE,
    "JobType" int NOT NULL,
    "JobObject" varchar(71) NOT NULL,
    "JobParameter" int NULL,
    "JobGroup" varchar(100) NULL,
    "Started" timestamp NULL,
    "Completed" timestamp NULL,
    "Status" int NULL,
    "Comment" text NULL
);

CREATE TABLE "_JobsParameters"(
    "JobParameter" bigint NOT NULL,
    "ParameterSet" int NOT NULL,
    "DimIdx" int NOT NULL,
    "ParameterValue" text NOT NULL,
    "Comment" text NULL
);

CREATE TABLE "_FactLoad"(
    "JobObject" varchar(50) NOT NULL UNIQUE,
    "CubeId" varchar(50) NOT NULL,
    "FactTable" text NULL,
    "FactChangesTable" text NULL,
    "FactLoadType" int NOT NULL,
    "ParameterId" int NOT NULL,
    "LastLoadedVersion" int NULL,
    "NewestVersionToLoad" int NULL,
    "SourceId" int NULL,
    "FactType" int NULL,
    "RelationalColumnName" varchar(50) NULL,
    "LoadFlags" bigint NULL,
    "Comment" text NULL
);

CREATE TABLE "_FactLoadParameters"(
    "Id" int NOT NULL,
    "DimIdx" int NOT NULL,
```

```
    "ColumnName" varchar(50) NULL,
    "TargetElement" varchar(71) NULL,
    "TargetElementUnknown" varchar(71) NULL,
    "TargetElementMissing" varchar(71) NULL,
    "Comment" text NULL
);

CREATE TABLE "_Hierarchies"(
    "HierarchyName" varchar(50) NOT NULL,
    "HierarchyDescription" varchar(150) NOT NULL,
    "DimensionName" varchar(50) NOT NULL,
    "SourceId" int NULL,
    "HierarchyType" int NOT NULL,
    "Param1" text NULL,
    "Param2" text NULL,
    "DefaultElement" varchar(71) NULL,
    "ExtendedProps" text NULL,
    "Comment" text NULL
);

CREATE TABLE "_Parameters"(
    "Id" bigint NOT NULL,
    "Key" varchar(50) NOT NULL,
    "DataType" int NOT NULL,
    "StringValue" text NULL,
    "IntegerValue" int NULL,
    "LinkId" bigint NULL,
    "Comment" text NULL,
    CONSTRAINT ID_KEY_PK PRIMARY KEY ("Id", "Key")
);

CREATE TABLE "_ScheduledJobs"(
    "Id" bigint NOT NULL UNIQUE,
    "JobType" int NOT NULL,
    "JobObject" varchar(71) NOT NULL,
    "JobParameter" int NULL,
    "Schedule" text NOT NULL,
    "Scheduled" int NOT NULL,
    "Disabled" int NOT NULL,
    "Comment" text NULL
);

CREATE TABLE "_Sources"(
    "SourceId" int NOT NULL,
    "SourceType" int NOT NULL,
    "ConnectionParameters" text NULL,
    "Comment" text NULL
);

CREATE TABLE "_Subsets"(
    "DimName" varchar(50) NOT NULL,
    "SubsetName" varchar(50) NOT NULL,
    "SubsetKind" int NOT NULL,
```

```
    "SubsetCategory" int NOT NULL,
    "SubsetTable" text NOT NULL,
    "Comment" text NULL
);

CREATE TABLE "_ErrorText"(
    "ErrorCode" int NOT NULL,
    "LanguageRegion" varchar(15) NOT NULL,
    "Text" text NULL
);

CREATE TABLE "_ErrorLogFactLoad"(
    "JobId" bigint NOT NULL,
    "ErrorCode" int NOT NULL,
    "ErrorDimIdx" int NULL,
    "ErrorDimName" varchar(50) NULL,
    "Element00" varchar(256) NULL,
    "Element01" varchar(256) NULL,
    "Element02" varchar(256) NULL,
    "Element03" varchar(256) NULL,
    "Element04" varchar(256) NULL,
    "Element05" varchar(256) NULL,
    "Element06" varchar(256) NULL,
    "Element07" varchar(256) NULL,
    "Element08" varchar(256) NULL,
    "Element09" varchar(256) NULL,
    "Element10" varchar(256) NULL,
    "Element11" varchar(256) NULL,
    "Element12" varchar(256) NULL,
    "Element13" varchar(256) NULL,
    "Element14" varchar(256) NULL,
    "Element15" varchar(256) NULL,
    "Element16" varchar(256) NULL,
    "Element17" varchar(256) NULL,
    "Element18" varchar(256) NULL,
    "Element19" varchar(256) NULL,
    "Element20" varchar(256) NULL,
    "Element21" varchar(256) NULL,
    "Element22" varchar(256) NULL,
    "Element23" varchar(256) NULL,
    "Element24" varchar(256) NULL,
    "Element25" varchar(256) NULL,
    "Element26" varchar(256) NULL,
    "Element27" varchar(256) NULL,
    "Element28" varchar(256) NULL,
    "Element29" varchar(256) NULL,
    "ElementCaption00" varchar(256) NULL,
    "ElementCaption01" varchar(256) NULL,
    "ElementCaption02" varchar(256) NULL,
    "ElementCaption03" varchar(256) NULL,
    "ElementCaption04" varchar(256) NULL,
    "ElementCaption05" varchar(256) NULL,
    "ElementCaption06" varchar(256) NULL,
```

```
    "ElementCaption07" varchar(256) NULL,
    "ElementCaption08" varchar(256) NULL,
    "ElementCaption09" varchar(256) NULL,
    "ElementCaption10" varchar(256) NULL,
    "ElementCaption11" varchar(256) NULL,
    "ElementCaption12" varchar(256) NULL,
    "ElementCaption13" varchar(256) NULL,
    "ElementCaption14" varchar(256) NULL,
    "ElementCaption15" varchar(256) NULL,
    "ElementCaption16" varchar(256) NULL,
    "ElementCaption17" varchar(256) NULL,
    "ElementCaption18" varchar(256) NULL,
    "ElementCaption19" varchar(256) NULL,
    "ElementCaption20" varchar(256) NULL,
    "ElementCaption21" varchar(256) NULL,
    "ElementCaption22" varchar(256) NULL,
    "ElementCaption23" varchar(256) NULL,
    "ElementCaption24" varchar(256) NULL,
    "ElementCaption25" varchar(256) NULL,
    "ElementCaption26" varchar(256) NULL,
    "ElementCaption27" varchar(256) NULL,
    "ElementCaption28" varchar(256) NULL,
    "ElementCaption29" varchar(256) NULL,
    "FactType" int NULL,
    "ValueNumeric" real NULL,
    "ValueString" text NULL,
    "Comment" text NULL
);

CREATE TABLE "_ErrorLogElementLoad"(
    "JobId" bigint NOT NULL,
    "ErrorCode" int NOT NULL,
    "Dimension" varchar(256) NULL,
    "DimensionCaption" varchar(256) NULL,
    "Hierarchy" varchar(256) NULL,
    "HierarchyCaption" varchar(256) NULL,
    "Element" varchar(256) NULL,
    "ElementCaption" varchar(256) NULL,
    "ElementType" varchar(256) NULL,
    "ParentElement" varchar(256) NULL,
    "ParentElementCaption" varchar(256) NULL,
    "Weight" real NULL,
    "Comment" text NULL
);

CREATE TABLE "_ErrorLogAttributeFill" (
    "JobId" bigint NOT NULL,
    "ErrorCode" int NOT NULL,
    "Dimension" varchar(256) NULL,
    "DimensionCaption" varchar(256) NULL,
    "Hierarchy" varchar(256) NULL,
    "HierarchyCaption" varchar(256) NULL,
    "Element" varchar(256) NULL,
```

```
    "ElementCaption" varchar(256) NULL,
    "TableId" int NULL,
    "AttributeFieldName" varchar(256) NULL,
    "AttributeDataValue" varchar(512) NULL,
    "AttributeFieldType" varchar(16) NULL,
    "AttributeFieldLength" int NULL,
    "AttributeFieldDecimals" int NULL,
    "Comment" text NULL
);

CREATE TABLE "_ErrorLogOther"(
    "JobId" bigint NOT NULL,
    "ErrorCode" int NOT NULL,
    "ObjectKind" varchar(256) NULL,
    "ObjectName" varchar(256) NULL,
    "ObjectCaption" varchar(256) NULL,
    "ErrorText" text NULL
);
```

Clear cube region example

These examples detail how the load from source process of clearing a cube region is set up. All examples use a FACTCUBE data cube with four dimensions: AAA, BBB, CCC, and DDD.

FACTCUBE	AAA
	BBB
	CCC
	DDD

This table lists the elements of the dimensions.

AAA	BBB	CCC	DDD
AAA00	BBB00	CCC00	DDD00
AAA01	BBB01	CCC01	DDD01
AAA02	BBB02	CCC02	DDD02
AAA03	BBB03	CCC03	DDD03
AAA04	BBB04	CCC04	DDD04

Cube clearing jobs

The jobs in the _Jobs table clear regions of FACTCUBE. This table lists the essential columns in the _Jobs table.

JobId	JobType	JobObject	JobParameter	Status
1	9	Clearout1	NULL	NULL
2	9	Clearout2	NULL	NULL
3	9	Clearout3	NULL	NULL
4	9	Clearout4	100	NULL

Each job has a job ID as normal for any job. Each also has a job type of 9 for clearing a cube region. The job object describes the details of the required job action of clearing out a region.

This table lists the details of the job objects in the _FactLoad table. Only the relevant columns are shown.

JobObject	CubeId	FactTable	FactChangesTable	FactLoadType	ParameterId
Clearout1	FACTCUBE	NULL	NULL	1	10
Clearout2	FACTCUBE	NULL	NULL	1	11
Clearout3	FACTCUBE	NULL	NULL	1	12
Clearout4	FACTCUBE	NULL	NULL	2	11

The four clear-out jobs: Clearout1, Clearout2, Clearout3, and Clearout4, are each described in detail. Each job clears a region of the FACTCUBE cube.

There are no source tables, the FactTable and FactChangesTable columns must be NULL. The FactLoadType column can be 1 for full load or 2 for partial load. A full load is a clear-out of every cell specified in the fact load parameters table. A partial load only clears out a subset of these according to the job parameters table.

The key to clearing out regions of a cube is in its dimensions and the latter's elements. These details are in the _FactLoadParameter table. The four clear-out processes are described in the three parameter IDs: 10, 11 and 12. Clearout2 and Clearout4 share a parameter ID.

ID	DimIdx	ColumnName	TargetElement
10	0	AAA	NULL
10	1	BBB	NULL
10	2	CCC	NULL
10	3	DDD	NULL
11	0	NULL	AAA00
11	1	BBB	NULL

ID	DimIdx	ColumnName	TargetElement
11	2	CCC	NULL
11	3	DDD	NULL
12	0	NULL	AAA00
12	1	NULL	BBB00
12	2	CCC	NULL
12	3	DDD	NULL

The fact load parameter is linked to the dimensions of the cube. The four clear-outs are detailed separately.

Clearout1

The _FactLoad table has the ID for Clearout1 as 10. There are four lines in _FactLoadParameters linked to ID 10. There is one for each of FACTCUBE's dimensions. These dimensions are numbered in accordance with the server's numbering of a cube's dimensions. They are zero-based.

The column name is actually a reference to the name of a column of a fact load table. The name is not needed here because clear-outs do not have source tables. However, the dimension it relates to can be put there for clarity.

In other words, the _FactLoadParameters table maps dimensions of the cube to dimensions (or part thereof) of elements defining the cells to be cleared out.

Consider the first row:

ID	DimIdx	ColumnName	TargetElement
10	0	AAA	NULL

This tells the process to use the whole of dimension 0, AAA, in the clearing out process.

All of them together for ID 10:

ID	DimIdx	ColumnName	TargetElement
10	0	AAA	NULL
10	1	BBB	NULL
10	2	CCC	NULL
10	3	DDD	NULL

Together they inform the process to use all the elements of all four dimensions. Clearout1 clears out the whole cube.

Clearout2

There is a small difference between the last process and Clearout2. The difference is that for dimension 0 (AAA), the column name is NULL and an element is named as target element.

ID	DimIdx	ColumnName	TargetElement
11	0	NULL	AAA00
11	1	BBB	NULL
11	2	CCC	NULL
11	3	DDD	NULL

This tells the process to clear out only the cells which have element AAA00 of dimension 0 (AAA) and any element of the other 3 dimensions. Clearout2 clears out a three-dimensional subcube.

Clearout3

The process can be further restricted. Clearout3 fixes dimensions 0 and 1 (AAA and BBB) and uses all elements of the other two dimensions.

ID	DimIdx	ColumnName	TargetElement
12	0	NULL	AAA00
12	1	NULL	BBB00
12	2	CCC	NULL
12	3	DDD	NULL

Clearout3 clears a two-dimensional subcube.

Clearout4

Referring to the _Jobs table once more, Clearout4 has a fact load type of 2. This is a partial load. A partial load restricts the load further than the fact load parameters allow. With the fact load parameters you can choose a whole dimension or a single element from it. But there is the facility to choose any elements from any dimension as well. For this, use the _JobsParameters table.

JobParameter	ParameterSet	DimIdx	ParameterValue
100	1	1	*
100	1	2	CCC03
100	1	3	DDD02,DDD03,DDD04

The JobParameter of 100 comes from the _Jobs table in a column of the same name.

The ParameterSet column groups clear-outs. You can choose any number for ParameterSet but it must be the same for each region to clear out.

The rows of the _ JobsParameters table are used together with the _FactLoadParameters table.

ID	DimIdx	ColumnName	TargetElement
11	0	NULL	AAA00
11	1	BBB	NULL
11	2	CCC	NULL
11	3	DDD	NULL

The first row of the _JobsParameters table refers to dimension 1 (BBB) and has a ParameterValue of *. This means use the whole dimension BBB. An asterisk (*) is a wildcard.

The second row instructs the process to restrict to element CCC03 of dimension 2 (CCC).

The third row tells the process to use only elements DDD02, DDD03, and DDD04 of dimension 3 (DDD).

A row for dimension 0 (AAA) must not be included because it already has a target element set in the _FactLoadParameters table.

As an alternative to using a wildcard (*) for ParameterValue, the row can be omitted. These _JobsParameters rows would achieve the same goal:

JobParameter	ParameterSet	DimIdx	ParameterValue
100	1	2	CCC03
100	1	3	DDD02,DDD03,DDD04

There is the option to define many regions in a single clear-out. This can be achieved by expanding the use of ParameterSet.

JobParameter	ParameterSet	DimIdx	ParameterValue
100	1	2	CCC03
100	1	3	DDD02,DDD03,DDD04
100	2	1	BBB02
100	2	2	CCC01,CCC04
100	2	3	DDD00,DDD01

This clears away two regions in the same clear-out, each defined by its ParameterSet value. The two regions cleared out are:

For ParameterSet 1:

- The element AAA00 of dimension 0 or AAA (defined in _FactLoadParameters)
- All elements of dimension 1 or BBB

- The element CCC03 of dimension 2 or CCC
- The elements DDD02, DDD03, and DDD04 of dimension 3 or DDD

For ParameterSet 2:

- Element AAA00 of dimension 0 or AAA (defined in _FactLoadParameters)
- Element BBB02 of dimension 1 or BBB
- Elements CCC01 and CCC04 of dimension 2 or CCC
- Elements DDD00 and DDD01 of dimension 3 or DDD

The two rectangular regions together are cleared out with the same job.

General technical reference

14

Limits of the OLAP Server

Cube limits

Item	Max. value	Comment
Number of dimensions in a cube	30	
How often can a dimension be used in a cube	1	
Number of cell notes in a cube	> 10.000.000	
Number of characters in cube name/description	50/150	
Number of characters in a text cell	1.000.000 bytes	
Number of rules in a cube	20 dims: ~ 2000 10 dims: ~ 4000 5 dims: ~ 8000	Varies by the complexity of the rules and the number of dimensions in the cube.
Number of accelerators in a cube	~ 32767	
Length of a cube rule	4-8 kByte Text	Depends on the complexity of the rule.
Length of a cube accelerator	499	
Length of a cube rule description	255	
Rule iteration depth	256	
Number of characters in a cell note	30000	
Number of values in a cube	2.000.000.000	Limited by available virtual memory.

Dimension limits

Item	Max. value	Comment
Number of elements in one dimension	10.000.000	
Number of children for one parent	10.000.000	
Number of parents for one child	84	
Number of consolidations in one dimension	~ 1.000.000	A dimension must have at least one base element.
Number of consolidation levels	127	
Number of factors in a consolidation	65.530	Number of children.
Number of factors in a dimension	> 1.000.000	
Number of dimensions	> 1.000	
Number of characters in dimension name/description	50/150	
Number of characters in element name	71	
Sum of the length of element names for a cell address	1440	
Number of rules in a dimension		Dimension rules are used like cube rules.

Subset limits

Item	Max. value	Comment
Number of elements in a subset	1.000.000	
Number of subsets (public and private)	> 100.000	The operating system limit of number of files in a directory can lower this limit.
Number of characters in a subset name	100	

Attribute limits

Item	Max. value	Comment
Number of attribute tables per database	> 3.000	
Number of attribute tables per dimension	3	

Item	Max. value	Comment
Number of characters in an attribute field short-name	50	
Number of characters in an attribute field long-name	50	
Number of characters in a text field of an attribute table	253	
Number of fields within an attribute table	49	

General limits

Item	Max. value	Comment
Number of connected databases	> 10	
Number of databases on one server	128	
Memory consumption for one database on a 64-bit Windows	2 TB (due to Windows limits)	Depends on the hardware and the operating system.

User limits

Item	Max. value	Comment
Number of user groups	> 10.000	
Number of users	> 10.000	
Number of users per group	> 10.000	
Number of groups per user	> 1.000	
Number of characters in a user name/description	50 / 200	
Number of characters in a user password	100	
Number of characters in a group name/description	50 / 200	
Number of concurrent connected users	16.000	

Level limits

Item	Max. value	Comment
Number of characters in a level name	50	

Object names

Starting with Infor PM OLAP 10.1 space characters are allowed in dimension and cube names, only leading and ending space characters are trimmed. Internally names are still compared case insensitive and without spaces, so “AB” is the same as “a b”. Also lowercase names can be created.

When updating an existing dimension, the internal name (case insensitive and without space characters) and the "external" name (case sensitive and with space characters) of the updated dimension must not differ in space characters.

Reserved characters for dimension, hierarchy, and cube names

These characters are not allowed in dimension, hierarchy, and cube names: `[ ] \ / : * ? " < > | <TAB> <LF> <CR>`

A hierarchy name cannot start with a number. This is reserved for future use. There is an exception, an implicit hierarchy can start with a number. An implicit hierarchy name is the same as the dimension name. This is allowed for backward compatibility.

Spaces in element names

OLAP Server before 10.5.x stripped all spaces (in front, in the middle, at the end of a string) from names before using the names in comparisons.

Starting with 10.6.0 only spaces at the beginning and at the end of a string are stripped. This behavior is not configurable.

Examples:

String	Stripped string
"ab"	"ab"
" ab"	"ab"
"ab "	"ab"
" ab "	"ab"
"a b"	"a b"
" a b "	"a b"
"a b"	"a b"

Reserved characters for element names

These characters are not allowed in element names: `[ ] <TAB> <LF> <CR>`

Reserved strings for element names

OLAP Server has some reserved strings that conflict with strings in other places if used as element names.

Starting with version 10.5.0 you cannot create elements with these names:

- B:*
- C:*

OLAP Server does not use or load existing dimensions that contain these element names.

In versions prior to 10.5.0, OLAP Server prints a warning into `AleaPR.txt` if these names are found. This warning states that these element names should not be used, but the dimension is loaded.

Reserved characters for database names

The following characters are not allowed in database names: `\ / | < > ? * " :`

File structure

File	Description
`Alea.ini`	Initialization file for OLAP Server and Communication Manager. Stores the database directory path, language settings etc. See "Alea.ini" on page 265)
`Mis.Alea.ComManager.exe`	OLAP Server Communication Manager executable.
`Mis.Alea.Service.exe` `Mis.Alea.Server64.exe`	Executable of OLAP Service (same as `Mis.Alea.ComManager.exe`, just as a service). OLAP Server executable.
`Db.ini`	OLAP Server configuration file for database settings There is a global `Db.ini` in the database root directory that applies to all databases and local ones

File	Description
	in the database folders that may override the settings in the global `Db.ini`. See "Db.ini" on page 271.
`mdsapi32.dll`	Graphical user interface features. These features supply several dialog boxes for OLAP Administration.
`Mis.Alea.Automation.dll`	COM API. Same functionality as VB(A) API, just as COM component.
`mdscp32.dll`	Provides client functionality. Deprecated, still there for compatibility reasons.
`Mis.Alea.ClientSupport.dll`	Provides client functionality.
`Mis.Alea.ClientUI.dll`	Provides client functionality.
`Mdsex.ini`	Configuration file mainly for Office Plus settings. Also used by OLAP Administration. See "Mdsex.ini" on page 286.
`mdsvba32.dll`	API for VB. Deprecated. Still there for compatibility reasons.
`Mis.Alea.VBAPI.dll`	API for VB.
`*.resources.dll`	Language resource files.
`Mis.Alea.ODE.dll`	For use with the ODE integration.
`Mis.Alea.EventAgentEngine.dll`	OLAP Server Event Server Connectivity
`*.tmp`	If you enable Undo for splashing operations, a `.tmp` file is created in the database folder with the necessary information for the undo.

Installed files and their locations

In the directory `C:\ProgramData\Infor\BI\OLAP` created during installation of a full standalone version:

`Alea.ini`

`Mdsex.ini`

In the directory `C:\ProgramData\Infor\BI\OLAP` created during installation of a server only:

`Alea.ini`

In the SYSTEM directory created during installation of a full standalone version (default = `C:\Program Files\Infor\BI\OLAP\bin64`):

`Createolapmetadata.sql`

`Createolapmetadataoracle.sql`

`Createolapmetadatapostgres.sql`

`Dbghelp64.dll`

`Iconv64.dll`

`Libeay64.dll`

`Mis.alea.clientsupport64.dll`

`Mis.alea.cosconnector64.dll`

`Mis.alea.language.resources.xml`

`Mis.alea.server64.exe`

`Mis.Alea.Server64.exe.config`

`Mpir64.dll`

`Provider`

`Ssleay64.dll`

`Zlib1_64.dll`

In the TUTOR directory created during installation of the English version of OLAP Server (default = `C:\Users\Public\Documents\Infor\BI\OLAP\Data\TUTOR_EN`):

`#__dim__.dim`

`#__grp__.dim`

`#__mds__.db_`

`#__tab__.dim`

`#_darea_.dbf`

`#_subset.dbf`

`#_tabacc.ttt`

`Actvsbud.dim`

`Alealog.txt`

`Aleapr.txt`

`Attrib.xls`

`Db.ini`

```
db_properties.xml
Editfor1.xls
Import.xls
LoginTrials
Measures.dim
Months.dim
Prodcode.dim
Products.d01
Products.dim
Products_ac.ttt
Products_ad.dim
Regions.dim
Totsales.ttt
Usedconfig
Years.ald
Years.dim
```

Using the initialization files

15

These `.ini` files are used for the configuration of Office Plus and the OLAP Server: `Alea.ini`, `Db.ini`, and `Mdsex.ini`.

Defining loadable databases using the .ini files

You can change the location of the LOCAL database root directory by editing your OLAP Server initialization file, `Alea.ini`. This file is located in the Windows directory. The path to the default location of the database root directory is `C:\Users\Public\Documents\Infor\BI\OLAP\Data`. The directory is created during installation.

When the LOCAL database root directory is located on a network drive, each new database is stored in a subdirectory of this database root. Therefore, the user must have network access rights to create subdirectories on that drive.

Each database must contain the suffix 'A' or 'M'. If a database/model name has the suffix 'A', the database is loaded automatically when OLAP Server starts, that is, specified cubes in the database/model are loaded into the memory.

See "Specifying cubes to loaded automatically" on page 263.

If the database name has the suffix 'M', the cubes in the database are loaded only when a client references the database.

Changing the path of the root and defining databases

1. Open the `Alea.ini` file and edit the parameter `DBRoot` in the `[INIT]` section. The path statement must be a valid path.
2. Edit the parameter for `DB` in the `[DATABASES]` section. Specify the subdirectory name only for each database to be loaded on this server. This subdirectory must be located under the directory specified in the `DBRoot` parameter.
3. After the database name, enter a comma (,), then the loading instructions.
4. Save the file and close it.

A typical entry in the `Alea.ini` might read:

```
[INIT]
DBROOT=C:\Users\Public\Documents\Infor\BI\OLAP\Data\
[Databases]
DEFAULT=
DB1=TUTOR,M
DB2=TUTOREA,M
DB3=GENESIS,M
```

In this example, the root directory is `C:\Users\Public\Documents\Infor\BI\OLAP\Data\`.

Note: `Alea.ini` uses the 8-character naming convention for the directory `..\Documents` and for every path containing a directory or subdirectory name that consists of more than 8 characters.

Two databases are defined as available for access. Both are specified to load cubes and dimensions manually.

Defining loadable databases for a remote server

Loadable databases for a remote server are defined in the same way as loadable databases for a LOCAL server. The entries must be made in the `Alea.ini` file.

Setting the default database

In the `[Databases]` section of `Alea.ini`, you can set a parameter for the entry 'Default'. The parameter is used to specify the default database for the OLAP Server. This entry guarantees backward compatibility for applications developed with earlier versions of OLAP Server. For example, when you set 'TUTOR' as the default database for the LOCAL server, all OLAP Server formulas or functions refer to the database TUTOR for data, if the server name is 'LOCAL' without a suffix.

Note: The default database must also be listed as a loadable database next to one of the `DBx` parameters. The default parameter specifies only which database to use as the default. It does not specify loading instructions. The default database parameter cannot be suffixed with the parameters 'A' or 'M'.

Example of correct usage:

```
[Databases]
DEFAULT=
DB1=TUTOR,M
DB2=TUTOREA,M
DB3=GENESIS,M
```

```
DB4=
DB5=
```

Specifying cubes to loaded automatically

When you specify the load instructions for an OLAP Server, you can instruct the server to load the database automatically. When databases are loaded, you can specify which of their cubes are loaded on start. You can choose to load all cubes, or only those specified in a list.

To specify cubes to be loaded automatically:

1 Open the `Db.ini` file and edit the parameter for `LOADTABLES` in the section `[INIT]`.

   To load all cubes, change the parameter to `ALL`. To load only specific cubes, change it to `LISTED`.

2 If you use `LISTED`, then also list each table to be loaded in the section [LOADTABLES]:

```
[LOADTABLES]
TABLE1=TOTSALES
TABLE2=
```

3 Save the file and close it.

Protocol used by the server

A protocol enables the client and the server to communicate in the Local Area Network (LAN). OLAP Server supports only TCP/IP, which must be enabled on the client to use Alea in the client–server mode.

Adjusting model properties in the .ini files

OLAP Server uses three different `.ini` files to manage settings for communications, database usage, Excel Integration options etc. These files are `MdsEx.ini` and `Alea.ini`, both located in your Windows directory, and `Db.ini` located in the directory of each OLAP Server database or the root directory (`..\DB Root`) of the database.

`Mdsex.ini` contains the Excel settings and parameters. All settings in this `.ini` file are specified in the Options dialog in Office Plus. The settings for each database (for example, whether SMP or Rules

caching is or is not used) are specified in the `Db.ini`. Settings for communications, locations of databases, default databases etc. are specified in the `Alea.ini`. The parameters for the `Db.ini` and the `Alea.ini` can be specified using the OLAP Administration.

The General component of OLAP Administration enables you to specify the general settings which are stored in the `Alea.ini`. The Database Settings component enables you to define your preferences for the databases. The parameters are set in the global `Db.ini` located in the root directory (`..\DB Root`) of the database, or in the `Db.ini` files of individual databases.

For an individual database, OLAP Server first looks at the settings in the `Db.ini` of that database. If a parameter is not set there, OLAP Server looks for the settings in the general `Db.ini`.

Loading client settings from the OLAP Server

In OLAP Server the client settings can be loaded from the OLAP Server using the `<Alea:Client Configuration>` tag in the file `db_properties.xml` located in the DBRoot directory. The settings defined in `db_properties.xml` are effective for all clients.

When a client connects to a server, it loads the configuration from the server, updates its `.ini` files and reloads the configuration. The name of the last connected server is stored and if you connect to the server again, the configuration is not loaded again.

We recommend that you have only one server where the client configuration is stored. If a client connects to 2 servers with client configuration, the result is undefined.

There are 2 modes for the configuration settings:

- Force: Overrides the client settings.
- Default: Adopts only settings from `db_properties.xml` which are not defined in the ini-file for the client.

The database XML properties can be edited using the XML request `Database PutProperties`, via the VBA function `ServerPutProperties` or by editing the `db_properties.xml` file. You can edit the file only if the database is not running.

Sample for `<Alea:ClientConfiguration>`:

```
<Alea:ClientConfiguration>
   <Alea:ConfigurationFile Name="MdsEx.ini">
      <Alea:Section Name="SETTINGS">
         <Alea:Key Name="SetManualCalc" Value="1" Mode="Force"/>
      </Alea:Section>
      <Alea:Section Name="SLICE">
         <Alea:Key Name="AutoCalc" Value="0" Mode="Default"/>
      </Alea:Section>
   </Alea:ConfigurationFile>
</Alea:ClientConfiguration>
```

Note: The `<Alea:ClientConfiguration>` tag can only be used once in the file db_properties.xml. The configuration may not take effect immediately. Sometimes you will have to recalculate the view or restart the client.

These ini-keys from `MdsEx.ini` are supported:

```
[SETTINGS]
SetManualCalc=1
[SLICE]
AutoCalc=1
```

Caution: Changing other ini-keys at runtime can have unexpected side effects.

Individual INI files

Each of the four `.ini` files is shown in a separate table. Their parameters are listed with their default settings, other acceptable values, and a brief description.

Alea.ini

The first initialization file for OLAP Server is `Alea.ini`, which is used to control server-specific settings. `Alea.ini` is located in `C:\ProgramData\Infor\BI\OLAP`.

Parameter	Required format	Acceptable values	Default value	Description
`[INIT]`				
The `[INIT]` section defines the OLAP Server settings, including where to find the files for operation.				
`ProgramRoot`	File system path			The root of executable files.
`DBRoot`	File system path			The root of database directories.
`LicenseRoot`				This parameter is obsolete starting with OLAP Server 10.5.0.
`Language`	Value	`001, 042, 049, 034, 033`	001	The country code for resources.

Parameter	Required format	Acceptable values	Default value	Description
`ShowLicExpiration`				This parameter is obsolete starting with OLAP Server 10.5.0.
`VISIBLE`	Value	Yes, No	No	If 'Yes', ComManager console is shown, even if the ComManager has been started by MDSCP. **Note:** If the Console Window of the Alea Server is displayed and the active Windows user is logged off, the Alea Server is shut down.
`DefaultStartupMode`	String	M (manual), A (automatic)	M	This key is used by OLAP Administration only. It defines the startup mode of new databases.

`BackupRoot`	File System Path		Empty	If defined, the server creates backups under this path.
				Structure:
				`BackupRoot`
				`Modelname`
				`Backup001`
				`Backup002`

`SystemMode`	Value	STANDALONE, CLUSTER, SOACLUSTER	STANDALONE	
`MinTLSVersion`	Value	10, 11, 12	12	Sets the minimum supported TLS (SSL) version to: 10: TLS 1.0 11: TLS 1.1 12: TLS 1.2 This disables older SSL protocols and also demands use of the highest available version.
`[COMMUNICATION]`				
The `[COMMUNICATION]` section defines the general communication settings.				
`Local`	Value	Yes, No	Yes	Enables local communication on present computer.

Parameter	Required format	Acceptable values	Default value	Description
TCPIP				This parameter is obsolete starting with OLAP Server 10.3.0.
Pack	Value	Yes, No	Yes	Packing of data transfer.
Crypt	Value	Yes, No	No	Encrypting data transfer.
EnhancedCrypt-ing	Value	Yes, No	No	Using enhanced encryption provider.
FQDN	Value	Yes, No	No	Using fully qualified domain name.
[COMMEXCHANGE]				
The [COMMEXCHANGE] section defines network visibility. Here you can specify what other OLAP Servers know about a running OLAP Server Communication Manager (CM), thus enabling a potential connection. Settings about network visibility of running ALEA network servers.				
SendTo	Value	All, None, Listed	All	None: do not broadcast. Listed : broadcast only to OLAP Server CMs listed in the parameter CM. All : broadcast to any computers where the OLAP Server CM is running.
CM...	Host name			List of computers where other aleacomms run.
[SRVREGISTER]				
The [SRVREGISTER] section defines the computers (Comm Managers) to which the current Comm Manager registers itself (see list box **Server Registration** in the Alea Administration). Settings concerning the registration of started Alea Servers				
CM...	Host name			List of other computers, where a OLAP Server CM may be running.
[CLIENTCONNECT]				
The [CLIENTCONNECT] section defines the behavior of the client in relation to communication. When both Local and TCP/IP are set to 'Yes' in the [COMMUNICATION] section, the OLAP Server first attempts to start the LOCAL server with the present CM, then using TCP/IP, it attempts to connect to the OLAP Server CMs listed under the CM parameters:				
CM…	Host name			List of computers where aleacomms may run and

Parameter	Required format	Acceptable values	Default value	Description
				which will be contacted for registration of started server.
[DATABASES]				
The [DATABASES] section defines which databases are available and how they are loaded.				
Default	Directory name		TUTOR	Default DB server on present computer.
DB…	Directory, A|M			List of DB servers which may be started on present computer. The options are: M: manual, or load the database when the first client makes a request. A: automatically, or load the database when the server is started.
[LOCAL]				
The [LOCAL] section specifies the location of the log file. This file is used to exchange information between the server, client and communication module on a LOCAL server. Local communication setting.				
LogFile	File name			Local “listen port” (memory file).
[TCPIP]				
The [TCPIP] section defines the TCP/IP settings for the LOCAL server. These settings are used in the event that the operating system is not able to return the IP information. Under normal circumstances, these settings do not need to be set. TCP communication settings.				
TCPAddr	IP address in dot format			IP address of present computer in dot format.
ListenPort	Digit		2904	TCP listen port of aleacomm on present computer.
SockBufSize	Digit		4096	The size of the internal socket buffers to send and receive data.
[UDPIP]				
The [UDPIP] section defines the listen port for the current computer. UDP communication settings.				
ListenPort	Digit		2904	The listen port on the present computer to ex-

Parameter	Required format	Acceptable values	Default value	Description
				change information between aleacomms.
[PROTOCOL]				
The [PROTOCOL] section defines the OLAP Server ComManager settings when writing messages.				
DisplayDate	Value	Yes, No	Yes	The messages contain the date.
DisplayTime	Value	Yes, No	Yes	The messages contain the time.
HighResTime	Value	Yes, No	No	The messages contain fractions of time in milliseconds.
PrintLevel	Value	Debug, Verbose, Quiet	Quiet	The level of producing the messages from aleacomm to its console window (if visible). These are the options: • Quiet: Error messages only. • Verbose: Error messages and warnings. • Debug: All messages
ProtLevel	Value	Debug, Verbose, Quiet	Quiet	The level of producing the messages from aleacomm to the protocol file commpr.txt , located in the same directory as the executable. The options are: • Quiet: Error messages only. • Verbose: Error messages and warnings. • Debug: All messages
MaxLines	Value	100 -1,000,000	1000	Maximum number of lines stored in commpr.txt.
Path	File system path		Directory where the binary alea-	Directory where commpr.txt is stored.

Parameter	Required format	Acceptable values	Default value	Description
			`comm.exe` resides.	
`EventLogLevel`	Value	Debug, Verbose, Quiet, None	None	Level of producing messages from aleacomm to the windows event log.
`[LICENSE]`				
This section is obsolete starting with OLAP Server 10.5.0.				
`[COS]`				
The `[COS]` section defines the Repository settings (User management configuration)				
`SSOProject`	string		Best_Practices_Templates	Name of repository project with defined permissions
`Repository`	string		Best_Practices_Templates	Name of repository that contains above project
`WinAuthenticationGuid`	string	guid string	`7f59033d-a4bd-4d98-9493-0886580ffe3d`	Guid of windows authentication system that will be used when performing ConnectWin
`[XMLA]`				
`StartDefaultCatalog`	Value	Yes, No	NO	YES: the XMLA provider automatically connects to the first available catalog, if there is no catalog specified in the connection string NO: only a list of catalogs is available, every other request returns an error. The modified settings take effect only after restarting the IIS.
`[CLUSTER]`				
`CentralDBRoot`	Value	Central DB Root		Name of the OLAP Server's cluster DB root. It is only read when the `SystemMode` key is set to `CLUSTER` in `Cluster.ini`.

Parameter	Required format	Acceptable values	Default value	Description
				See "Cluster.ini" on page 294.
[HTTP]				
ListenPort	Value	Integer	8210	The port for the unsecured communication to work on.
SecurePort	Value	Integer	8211	The port for the secured communication to work on.
RootCertificate	file name		root.pem	Truststore, certificate file in pem format.
Certificate	file name		server.pem	Certificate file in pem format containing the server certificate and the associated private key. The private key can be stored encrypted.
PassFile	file name		pass	Password file with encrypted password to decrypt the private key in server.pem. The password can be set via OLAP Server console.

Db.ini

The second initialization file is the `Db.ini` file. It is used to define model-specific settings. This file is located in the database directory, or in the database root directory. OLAP Server first looks at the database directory for the file. If the file is not there, it looks at the database root directory. If a `Db.ini` file exists in a database directory, the settings apply to that database. When the file is located only in the database root, the settings apply to all databases located in the root directory.

Parameters marked by * are re-read during server operation by calling the VB function `ReReadINIFiles`

See VB SDK.

Parameter	Required format	Acceptable values	Default value	Description
[INIT]				
The [INIT] section defines the general server settings.				
SMP	Value	ASYNC, SYNC	ASYNC	This parameter is obsolete starting with Infor PM OLAP 10.4.0.
NetServer	Value	YES, NO	NO	This server is a network server.

Parameter	Required format	Acceptable values	Default value	Description
`LoadTables`	Value	ALL, NONE, LISTED	NONE	Load specified tables immediately after the start of the server.
`Scheduler`*	Value	YES, NO	YES	Enables execution of scheduled events of the server.
`MultipleLogin`	Value	YES, NO	YES	Enables multiple connections with one user name.
`DefaultHandling`*	Value	NONE, READ, WRITE, ALL	ALL	Sets the default access rights to cube or elements, if table access-restriction table ("TABACC") or Dimension Access Control cube (’DAC’) is not defined.
`UseCallback`*	Value	YES, NO	YES	Enables sending callback information through network to client. `Mdscp32.dll` uses callback to show client-side progress bar while loading the table.
`EnableEvents`	Value	YES, NO	NO	Enables event handling on the server. The value is set to Yes, if the Startup Mode is set to Automatic + Events or Manual + Events..
`ListenPort`	Digit		Dynamic assignment by aleacomm	Sets a specific TCP/IP-port for communication. Especially needed for Java-Client.
`ProcessorCount`	Value	1,…	9999	Set this parameter to the number of processors on your computer for minor improvement of multi-processor-support. Even with default settings Alea works well on a multi-processor computer.
`MaxCalcTime`*	Value	1 - 3600	3600	Sets the number of seconds that a read-process may be calculated in the kernel, before it is considered a ’long-request’ that may be stopped if there is a pending write-request.
`SAPHierarchyPrefix`				This parameter is obsolete starting with Infor PM OLAP 10.0

Parameter	Required format	Acceptable values	Default value	Description
`ProcessAffinityMask`	Hexadecimal value	`<0x00000001, 0xFFFFFFFF>`	Current process affinity	Set this value & System affinity as Process Affinity.
`UndoFilesRoot`	File system path		Database directory	Directory for undo files created during splashing.
`VISIBLE`	Value	YES, NO	NO	If 'YES', the server console is shown, even if the server is started by COMM Manager.
`OldRulesEditor`				This parameter is obsolete starting with Infor PM OLAP 10.0.
`MaximumSplashValues`	Value	`<0, 5000000>`	100.000	Maximum number of basic cells, where a value can be written during one splashing operation. Used for 'Equal allocation mode' only.
`AllowSplashOnRuleCells`	Value	YES, NO	NO	If this is 'NO', the algorithm does not allow rule-calculations in cells changed by the algorithm or in the external source cell (if used). If this is enabled, the algorithm does not check for rule-calculated cells. Due to this the calculated value may differ from the one entered by the user.
`MaxTileSize`*	Value	`<1e1,1e100>`	1E80	Maximum number of cells used for internal batch rule-calculation. Use 1eX form to avoid problems with decimal separator.
`MaxExportTileSize`*	Value	`<1e1,1e100>`	MaxTileSize	Maximum number of cells in batch when data area is processed. It is also used to control the fragment size of a cube used in the engine verificator feature.
`OldRulesEngine`*	String	NEVER, ALWAYS, ACCELERATED	ACCELERATED	Enables old rules calculation engine for all, none, or accelerated cubes. Using the enhanced rules engine may be faster for non-accelerated rules. See *OLAP Server Rules Engine Guide*.

Parameter	Required format	Acceptable values	Default value	Description
`TestCodePage`	Value	YES, NO	YES	Server logon is allowed only to clients with code pages identical to the code page of the server
`InvalidAccelLimit`	Value	`<0, 2147483647>`	1000	If the count of invalid accelerator flags in a cube is higher than `InvalidAccelLimit`, the cube is indexed again. Otherwise the cube is not indexed and only the invalid accelerator flags are removed.
`CheckRightsOnC`	Value	YES, NO	NO	YES: If an event is generated for error 6 (KE_BADTYPE) during event generation, an additional test for user rights is made. If the result of this test is 34 (KE_NO_RIGHTS), 34 is added to the event instead of 6, otherwise error 6 remains there.
`ShowDetailedErrorMessagesInAleaSnapin`	Value	YES, NO	NO	Sets the type of error-message shown in Alea Administration. YES: The error-message can contain more information, like error-number or name of the procedure, where the error was caught.
`AcceleratorsRequired`*	Value	YES, NO	NO	Rules on basic cells are evaluated only if the cell is correctly accelerated.
`UserFunctionCompatibility`	Value	YES, NO	NO	If set to YES it affects these functions: • mdsGetGroupId: KE_NO_RIGHTS is returned in Uni.lError. • mdsGetGroupCount KE_NO_RIGHTS is returned in Uni.lError. • mdsGetUserCount returns 1 Uni.lCount • mdsGetUserId

Parameter	Required format	Acceptable values	Default value	Description
				If Uni.IId is 1 it behaves the same way as if Uni.IId is set to MDS_CURRENT_USER • If set to NO functions mentioned before return KE_OBSOLETE_FUNCTION
`AllowedUserAtStartup`	String			Name of user that can connect even during reservation mode waiting for the EventAgent startup.
`AttributesCaseSensitive`	Value	YES, NO	NO	YES: Attribute value tests are case sensitive (default until 5.2.0). NO: Attribute value tests are not case sensitive. Applies only to searching in dimension browser and attribute driven subsets.
`LicenseRetrievalDelay`				This parameter is obsolete starting with OLAP Server 10.3.0.
`AllowReferenceToInaccessibleMembers`	Value	YES, NO	NO	Allow reference to inaccessible members. If Yes, MDX statements can contain references to inaccessible members.
`MulticoreCalc`	Value	YES, NO	YES	Split a request into a number of parallel processed tasks (CSISweep).
`MaximumRequestCells`	Value	`100000, 2147483647`	10000000	The maximum number of requested cells per request that is used in MDX engine.
`MulticoreUserLimit`	Value	<0, n>	0	The maximum number of users/requests in the threadpool they can calculate in parallel. Over this limit the request will be calculated as single core calculation.
`RelationalConnectionString`		String value		Connection string formatted according to DBMS requirement. The target OLAP database

Parameter	Required format	Acceptable values	Default value	Description
				MY_DB should be replaced with one the user wants to load. The default value is Relational-ConnectionString= Driver={SQL Server Native Client 10.0}; Server=localhost\sql2008; Database=MY_DB; Trusted_Connection=yes; Pooling=false;
`[MEMORY]`				
The `[MEMORY]` section defines the memory settings for the server.				
`DimensionPage`				This parameter is obsolete starting with Infor PM OLAP 10.0.
`StringPage`				This parameter is obsolete starting with Infor PM OLAP 10.0.
`CheckTables`*	Value, combination of values	L, S, M, or any combination thereof O: this value is obsolete starting with Infor PM OLAP 10.0.		Specifies when the cube integrity is checked. This check compares the cube in memory to the cube on disk. If inconsistencies are found, they are repaired automatically. • L: upon loading • S: upon saving • M: manual
`CacheSize`*	Value	`<0, 100000>`	1	Cache for calculated values (rules calculated and consolidated values). '0' disables the cache. Any value greater than 0 enables the cache.
`UseTrefIndices`				This parameter is obsolete starting with Infor PM OLAP 10.0.
`AnswerSize`				This parameter is obsolete starting with Infor PM OLAP 10.0.
`CSDSlots`	Number	`<256,16384>`	1024	Number of slots used for calculation - each calculated slice needs one slot. Each rule cell-reference needs one. This number limits the amount of request calculated simultaneously and recursion depth. Currently 1 slot repre-

Parameter	Required format	Acceptable values	Default value	Description
				sents about 10kB of computer memory.
[LOADTABLES]				
The [LOADTABLES] section defines which cubes are loaded, when the server is started.				
Table1	Cube name			Name of a server table (cube) loaded into memory automatically when the server is started. Requires that the parameter 'LoadTables' in the [INIT] section be set to 'YES', and that the database name (in the [DB]) section of the Alea.ini be parameterized with 'A'.
[SCHEDULER]				
The [SCHEDULER] section defines the scheduled events of the server.				
SaveServer*	Value	YES, NO	YES	Server performs periodical saving of data.
FirstSave*	hh:mm	<00:01,23:59>, NOW	15:00	Sets the first time a server will automatically be saved in the 24-hour format.
SaveInterval*	hh:mm	00:01-23:59	1:00	Sets the saving interval in hours, for the server. In this case, save every hour.
UserWatchDog*	Value	YES, NO	NO	Checks for inactive logged users. Enables checking for inactive users on the server.
UserTimeOut*	hh:mm	00:01-23:59	1:00	Maximum time a user can remain logged onto a server without a request. Requires that parameter UserWatchDog be set to 'YES'. If this time is exceeded, the user will be logged off from the server automatically.
CommandFile*	Value	YES, NO	YES	Checks for the aleacommand file every 5 seconds. If it exists, commands specified in the file are performed.
ACMEImporter*	Value	YES, NO	NO	Checks for the importstart file every 5 seconds. If it exists, the data is imported.

Parameter	Required format	Acceptable values	Default value	Description
`Check*`	Value	YES, NO	NO	Reports present state of the SMP subsystem every 2 seconds. Visible console window expected or PROTOCOL-ProtLevel with value DEBUG to save these messages in `Aleapr.txt`.
`BackupServer`	Value	YES, NO	NO	Server performs periodical backup of data.
`FirstBackup`	hh:mm	`<00:01, 23:59>, NOW`	15:00	Time of first backup.
`BackupInterval`	hh:mm	`<00:01, 23:59>`	01:00	Backup interval.
`BackupPath`	Path		`<database directory>\BACKUP`	Backup path.
`BackupNumbering`	Value	YES, NO	NO	YES: the suffix 'nnnn' is attached to the backup path to keep multiple backup directories. 'nnnn' is the number stored in BackupNextNumber with leading zeros.
`BackupNextNumber`	Value	`<0,9999>`	0	Next backup number. The server increases the backup number after every backup, no matter if the backup operation was scheduled or manually started.
`BackupMaxNumber`	Value	`<-1,9999>`	-1	If the backup number exceeds the specified maximum backup number, the Next backup number is set to 0. If -1 is set, the backup number is unlimited.
`EmptyBackupTargetDir`	Value	YES, NO	YES	Deletes the content of the backup directory. **Caution:** If set to YES , the content of the backup directory is deleted before each backup.

Parameter	Required format	Acceptable values	Default value	Description
`CellToSQL`	Value	YES, NO	NO	Checks for records in `Alealog.txt` every 10s and transfers them to SQL Server. **Caution:** Splashing information in `Alealog.txt` will not be written into the relational database.
`CellToSQLConnectionString`	Value		`Server=localhost; Trusted_Connection=yes; Database=Alealog;`	SQL Server connection string. See "Connection Strings" on page 22.
`CellToSQLTable`	Value		Alealog	Name of the SQL Server table.
`CellToSQLOperationsTable`	Value		AlealogOperation	SQL Server table name for operations in version 2.
`CellToSQLFireTriggers`	Value	Yes, No	No	Obsolete starting with OLAP Server 10.4.0.
`CellToSQLOverwriteSettings`	Value	Yes, No	Yes	YES: `LOG\LoggedOperations` setting is not overwritten to ALL.
`DBLoad`	Value	Yes, No	No	Check regularly for relational jobs.
`DBLoadInterval`	Value	`<1,3600>`	60	Polling interval for `DBLoad` tasks in seconds.
`UnloadDBOnIdle`	Values	Yes, No	No	Checks if no users are connected and unloads the database from memory after a time-out. All cubes and dimensions are saved if necessary.
`UnloadDBIdleTimeout`	Values	>=1	180	Specifies the idle time (no user is connected) of the database, in minutes, before it is unloaded from memory.
`Fire SQL triggers` *	Values	Yes, No	No	This parameter is obsolete starting with Infor BI OLAP Server 10.4.0.

Parameter	Required format	Acceptable values	Default value	Description
SQL: Overwrite Definition of the cubes *	Value	Yes, No	Yes	If set to Yes , the cell changes of all cubes are transferred to SQL Server, regardless of the setting in Definition of the cubes.
[LOG]				
The [LOG] section defines the server protocol operations.				
LogFile				This parameter is obsolete starting with Infor PM OLAP 10.
LogCount				This parameter is obsolete starting with Infor PM OLAP 10.
LoggedOperations*	Value	NONE, TRANS, ALL	NONE	Operations logged in the Alealog.txt file.
LogIPAddress*		This parameter is obsolete starting with Infor PM OLAP 10.0 Internally LogIPAddress is set to YES. Alealog.txt always contains the IP address of the user.		
LogXML *	Value	YES, NO	NO	For tracing Alea XML API calls (database root directory).
RequestLog *	Value	YES, NO	NO	Complete request log file for each user.
RequestLogBuffer *	Value	<1, 10000>	1	Number of requests written to requestlog in one write operation. The optimal value in a multiuser environment is between 100 and 500.
RequestLogResponses *	Value	YES, NO	NO	When logging requests, log responses too. Dependent on Db.ini\LOG\RequestLog.
Transactions *	String	NONE, TRANS, ALL	NONE	Specifies wha Alealog.txt cell changes are written to Alealog.txt. • NONE - no changes • ALL - all changes of all cubes • TRANS - only changes on cubes with transactions activated - this switch depends

Parameter	Required format	Acceptable values	Default value	Description
				on Transactions Key in this section When the value is BASIC - only manually entered values and imported are written to `Alea-log.txt`.
`LogVersion`	Value	`<1,2,3,4,5,6>`	1	Logging version: 1: Only changed cells are logged 2: Additional information about the cell and splash operations are logged (table AlealogOperations) 3: IPv6 4: 30 dimensions To use ExportAleaLog with a relational database with 30 dimensions you must use LogVersion=4 5: Includes fields for hierarchies 6: Increased string cell limit (1,000,000 bytes)
`LogSplash` *	Value	YES, NO	NO	Logs two additional lines into `Alealog.txt` for splashing requests.
`LogTimeInUTC`	Value	YES, NO	NO	Logs with UTC time.
`LogSettingsOnRead`	Value	YES, NO	NO	Logs the configurations used into the standard logs: `Aleapr.txt` , console window and Windows event log. The content is independent of the log level (Quite, Verbose or Debug). Passwords are replaced by the fixed string 'xxx', regardless of their value and length.
`[PROTOCOL]`				
The `[PROTOCOL]` section defines the OLAP Server CM setting when writing messages.				
`DisplayDate`*	Value	YES, NO	YES	The messages contain the date.
`DisplayTime`*	Value	YES, NO	YES	The messages contain the time.

Parameter	Required format	Acceptable values	Default value	Description
`HighResTime`*	Value	YES, NO	NO	The messages contain the fraction of time in milliseconds.
`SysLevel`*	Value	DEBUG, VERBOSE, QUIET	QUIET	The level of producing the messages from this server to its console window (if visible). The options are: • QUIET: write error messages only. • VERBOSE: write error messages plus warnings. • DEBUG: write all messages to the event log.
`ProtLevel`*	Value	DEBUG, VERBOSE, QUIET	QUIET	Level of the messages from this server to the log file `Aleapr.txt`, located in the same directory as the executable. These are the options: • QUIET: write error messages only. • VERBOSE: write error messages plus warnings. • DEBUG: write all messages to the event log.
`MaxLines`	Value	`<100,1000000>`	1000	Maximum number of lines to store in the protocol file (`Commpr.txt`).
`EventLogLevel`	Value	DEBUG, VERBOSE, QUIET, NONE	NONE	The level of producing the messages from this server to the Windows event-log.
`RuleDebug`*	Value	<0,5>	0	Debug level of the rules engine • 0: disabled • 1, 2: all rules are logged • 3, 4: only rules taking more than a second. ProtLevel/SysLevel must be set to DEBUG.
`[Caching]`				
The `[Caching]` section describes the general server settings.				

Parameter	Required format	Acceptable values	Default value	Description
`<Cube name>`	String	USE, DONTUSE	USE	Use/Do not use cache for the specified cube.
`[ExecuteCommand]`				
The `[ExecuteCommand]` section executes commands.				
`Startup...`	Command string			These commands are sent to the OS at the start of the server. They may contain parameters.
`[COS]`				
The `[COS]` section defines the Common Objects Store configuration				
`ProjectName`	String	String	Best_Practices_Templates	Name of repository project with defined permissions
`ModelName`	String		<DATABASE>	Model name for identification in Repository.
`Repository`	String		Best_Practices_Templates	Name of repository that contains above project
`[EventAgent]`				
The `[EventAgent]` section describes the Event Agent configuration.				
`SendNetMessageOnError`	String	TRUE, FALSE	FALSE	Runtime and user-defined errors may be issued through the API function `ErrorExecute()` as screen messages.
`PushRuleIgnoreUser`	String			Determines which user is ignored by the Event Agent.
`StartUp`	String	TRUE, FALSE	FALSE	Determines if the Event Agent is started automatically by the Alea Server.
`SendNetMessage`	String	TRUE, FALSE	FALSE	Within the Event Agent screen, messages may be issued through the API function `SetNetMessage()`.
`PRSLogFile`	String	TRUE, FALSE	TRUE	
`PRSMaxLogEntries`			1000	Max. number of log entries in the log file.
`EventFilterCube`	String			Name of the trigger cube.

Parameter	Required format	Acceptable values	Default value	Description
`OnCellEvent`	String	ON, OFF	ON	ON: OnCellChange events are processed.
`OnDimensionEvent`	String	ON, OFF	ON	ON: OnDimensionChange events are processed.
`OnAttributeEvent`	String	ON, OFF	ON	ON: OnAttributelChange events are processed.
`OnGenericEvent`	String	ON, OFF	ON	ON: OnGeneric events are processed.
`OnProtocolError Event`	String	ON, OFF	ON	ON: OnProtocolError events are processed.
`PRSProtLevel`	String	QUIET, VERBOSE, DEBUG	QUIET	The level of producing the messages from Event Agent to the log file.
`[PROVIDER]`				
`UniqueNameFormat`	Value	LONG, SHORT, REDUCED	LONG	Long or short format of member unique name.
`RemoveDuplicates InAleaSubsets`	Value	YES, NO	NO	YES: Converting OLAP subsets to Named Sets only the first OLAP element is used.
`[HTTP]`				
`ListenPort`	Value	Integer	8210	The port for the unsecured communication to work on.
`SecurePort`	Value	Integer	8211	The port for the secured communication to work on.
`SessionTimeOut`*	Value	60...5400	900	Length of timeout for HTTP(S) based connections in seconds.
`[CLUSTER]`				
`ClusterEnabled`	Value	YES, NO	NO	Specifies if the database works in cluster mode. • YES enables the cluster mode for database. • NO disables the cluster mode. • Any value other than YES is treated as NO.

Parameter	Required format	Acceptable values	Default value	Description
`UpdatingNode`	Value	Computer name that is the Com-Manager computer name	Empty	Specifies the node on which the data is saved (all other nodes are read-only).

`Alealog.txt` file

	Transactions=BASIC	Transactions=ALL
LoggedOperations=NONE	No value is logged in `Alealog.txt`	No value is logged in `Alealog.txt`
LoggedOperations=TRANS	Basic cells changed manually or imported to cubes with activated transactions are logged in `Alealog.txt`	All changes of basic cells of cubes with activated transactions are logged in `Alealog.txt`
LoggedOperations=ALL	Basic cells changed manually or imported are logged in `Alealog.txt`	All changes of basic cells are logged in `Alealog.txt`

Testing the Code Page

1 Server logon is allowed only to clients with code pages identical to the code page of the server.

   It depends on the language settings if computers can work together. Strings are saved on a server in ANSI form in codepage of the host operating system. When data is transferred to a client computer with a different codepage, it can be damaged if it contains characters which are not supported by the codepage of the OS running on the client computer. To prevent such damage data communication between computers with different codepages is not allowed.

   Because old client applications do not send their actual codepage it is assumed they have a codepage different to the server. If the administrator knows that there are no codepage specific data in the model (for example, only English text in a model running on a Chinese OS) they can change the TestCodePage setting to NO and the old clients can logon.

   User name and IP address of any rejected user can be found in the LoginTrials protocol file.

2 The server only loads a cube/dimension into the memory if the code page of the cube/dimension is identical to the code page of the server.

   All Alea `*.ttt` and `*.dim` files contain the code-page information of the server that saved these files. When the server starts, all `*.ttt` and `*.dim` files are scanned if they have been saved in a code page different from the active one.

Mdsex.ini

Another initialization file for use with OLAP Server is `Mdsex.ini`. It is used to define user preferences for Office Plus. This file is located in the directory `C:\Users\UserName\AppData\Roaming\Infor BI\OLAP` (`%APPDATA%\Infor\BI\OLAP`). If there is no profile available for the user, the file is stored in `C:\ProgramData\Application Data\Infor\BI\OLAP` (`%ALLUSERSPROFILE%\Application Data\InforBI\OLAP`).

Parameter	Required format	Acceptable values	Default value	Description
`[INIT]`				
This section is stored in the `Alea.ini` file.				
`DBROOT`	Text	Private view path		
`Language`	Number	Alea language		
`LicenseRoot`				This parameter is obsolete starting with OLAP Server 10.5.0.
`HelpFile`	Text	Path to helpfile		
`[DATABASE]`				
`DEFAULT`	Text	Reading default DB		Used by `mds.xla`
`[TRANSFER]`				
The `[TRANSFER]` section defines the path settings for directories specifically for use with OLAP Server.				
`Import`	File path	Import directory	`C:\Users\Public\Documents\Infor\BI\OLAP\Transfer`	Destination directory for import files.
`Export`	File path	Export directory	`C:\Users\Public\Documents\Infor\BI\OLAP\Transfer`	Destination directory for export files.
`Template`	File path	Template directory	`C:\Documents and Settings\All Users\Documents\Infor\BI\`	Directory for saved templates for use with the Alea browser.

Parameter	Required format	Acceptable values	Default value	Description
			`OLAP\Tem-plate\`	
`View`	File path	View directory	`C:\Docu-ments and Settings\All Users\Documents\Infor\BI\OLAP\View\`	Directory for saved views. This can either be a local directory for private views, or a network directory for public views.
`DefaultView`	File path	Path to the view directory See "Path format of a DefaultView" on page 293.	empty (there is no path to a DefaultView defined)	If a value for this parameter is set, a custom default view is searched in this directory. Search order for a default view: 1. private view location 2. DefaultView location 3. public view location **Note:** You cannot load or save views to this path, you must copy a default view there manually.
`Errorlog`	File path and file name	Log file	`C:\Users\Public\Docu-ments\In-for\BI\OLAP\Trans-fer\Error.log`	Directory and file name of error log.
`[SETTINGS]`				
The `[SETTINGS]` section defines the preferences for the Office Plus				
`SaveLayout`	Value	0, 1	0	Saves the layout of the attribute tables: • 0: do not save. • 1: save.
`Server`	Name	Any valid server name	Local	Primary server

Parameter	Required format	Acceptable values	Default value	Description
`Login`	Value	0, 1	0	Automatic log-on at start: • 0: do not automatically log-on • 1: automatically log-on to server
`UserLevel`	Value	1, 2, 3	2	User levels: • 1: Beginner • 2: Advanced • 3: Professional
`SaveOnExit`	Value	0, 1	0	Save on log-out from local server: • 0: do not save • 1: save
`BrowserEntry`	Value	0, 1	1	Dimension browser
`SetManualCalc`	Value	0, 1	1	Activate Excel's manual recalculation at startup: • 0: do not set calculation to manual • 1: set calculation to manual
`TM1comLayer`	Value	0, 1	0	• 0: do not use Comm layer • 1: use Comm layer
`ShowNullValuesAsNA`	Value	0, 1	0	Show empty cells as #NA (DBGETx): • 0: do not show null values as 'N/A' • 1: show empty values as 'N/A'
`NA`	Text	Text string	None	Show '#N/A'
`OLDNA`	Text			

Parameter	Required format	Acceptable values	Default value	Description
`VersionWarning`	Value	0, 1	0	Checks Alea executable files for compatibility: • 0: do not check the files • 1: check the files
`Version`	Version number			
`ShowErrorMsg`	Value	0, 1	0	Shows a dialog to make a choice displaying current or all following errors.
`ShowSplash`	Value	0, 1	1	Enables or disables splash screen on Alea startup.
`[SLICE]`				
The `[SLICE]` section defines preferences for the OLAP Server Cube Browser.				
`FileTemplate`	File path and file name		`C:\Documents and Settings\All Users\Documents\Infor\BI\OLAP\Template\MIS.XLT`	Defines path and file of the browser template.
`UseTemplate`	Value	0, 1	1	Use template: • 0: do not use the template • 1: use the template
`Border`	Value	0, 1	0	Show frames: • 0: do not show borders • 1: show borders
`Header`	Value	0, 1	0	Show row and column headers: • 0: do not show headers • 1: show headers

Parameter	Required format	Acceptable values	Default value	Description
`ColumnWidth`	Value	1 - 99	10	Column width
`FormularType`	Value	1, 2, 3	1	Conversion to formula view: • 1: use DBGETC • 2: use DBGET • 3: use Values
`AutoCalc`	Value	0, 1	1	Automatic recalculation: • 0: do not set calculation to automatic • 1: set calculation to automatic
`AutoResizeRowTitle`	Value	0, 1	1	Automatic resize of row titles.
`ShowNotes`	Value	0, 1	0	Mark cells with notes.
`ShowNullValuesAsNA`	Value	0, 1	0	Show empty cells as #N/A: • 0: do not show null values as 'N/A' • 1: show null values as 'N/A'
`Indent`	Value	0 - 10	2	Automatic indentation.
`NumberFormat`	XL number format		Standard	Number format.
`FormulaType`	Value	1, 2, 3	1	DB Get formula-type: • 1: DBGETC formulas are used for slice generation • 2: DBGET formulas are used for slice generation • 3: slices are generated with values instead of formulas
`MaxCellCount`	Value	`<1, 16711425>`	2000	Max. number of cells in a view.
`SaveViewWithoutValues`	Value	0, 1	0	When saving a view the values in view data will not be visible.

Parameter	Required format	Acceptable values	Default value	Description
`WarningMode`	Value	0, 1	0	Detailed error on view structure changes (element error).
`EnableStructureChangedWarning`	Value	0, 1	1	Displays warning that the structure of the view has changed.
`AddSliceElementCount`	Value	0 - 250.000	16000	If element count of a dimension is greater than this value, a new algorithm is used when loading views (*.alv).
`[SYMBOL]`				
The `[SYMBOL]` section defines the Alea tool bar settings.				
`Enable`	Value	0, 1	1	• 0: do not display the tool bar • 1: display the tool bar
`[LOCATION]`				
The `[LOCATION]` section defines the locations of the Alea files.				
`XLA`	File path		`XLA=C:\Program Files (x86)\Infor\BI\OfficePlus\bin`	Path to `MDS.XLA`.
`[EXCEL]`				
Location of Excel executable				
`1=`	File path and file name			Location of Excel executable.
`[HISTORY]`				
The `[HISTORY]` section defines the history of the FindItemDialog.				
`History0`	Text	Text string	None	
`History1`	Text	Text string	None	
`History2`	Text	Text string	None	
`History3`	Text	Text string	None	
`History4`	Text	Text string	None	

Parameter	Required format	Acceptable values	Default value	Description
History5	Text	Text string	None	
History6	Text	Text string	None	
History7	Text	Text string	None	
History8	Text	Text string	None	
History9	Text	Text string	None	
History10	Text	Text string	None	
History11	Text	Text string	None	
History12	Text	Text string	None	
History13	Text	Text string	None	
History14	Text	Text string	None	
History15	Text	Text string	None	
[BUFFER]				
The [BUFFER] section defines the cache settings of a formula slice.				
DestroyBufferAfterDisconnect	Value	0, 1	1	When the buffer loops it checks, whether the connection still exists. If not, it destroys the buffer of the values.
[DEBUG]				
The [DEBUG] section defines the settings for the client side protocol. The information is stored in the XiProtocol_2844.log file, where the number is the process ID of the client. The file is located in the DBRoot directory. XiProtocol_2844.log is a text file, records are separated by a line break, each record can have multiple columns separated by TAB The logging works for mdsex32dll, mdsvba32.dll, and mdsaut32.dll. The log file is only created if a log event occurs.				
ProtocolVB-Calls	Value	0, 1	0	Logs calls to all VB API functions.
ProtocolXL-Calls	Value	0, 1	0	Logs calls to all Alea EXCEL functions.
ProtocolStartUp	Value	0, 1	0	Logs startup actions/initialization.
ProtocolError	Value	0, 1	0	Logs errors from: • VB functions (all)

Parameter	Required format	Acceptable values	Default value	Description
				• EXCEL functions (most of them) • view and GUI errors
`[DIALOGPLACEMENT]`				
The section `[DIALOGPLACEMENT]` is used to store the last position and size of the client windows.				
`[SPLASHER]`				
The `[HISTORY]` section contains the Splasher configuration.				
`Version`	Text			
`Enabled`	Value	0, 1	0	Enables Splasher.
`Calculate`	Value	0,1; meaning reversed	0	Recalculate sheet after splashing operation.
`UserDialog`	Value	0,1; meaning reversed	0	
`Connect`	Value	0, 1	0	
`ErrorCorrection`	Value	0, 1	0	Correction of rounding errors.
`Rounding`	Value	0, 1	0	
`DecimalPlaces`	Value		6	Number of decimal places used for the calculation (maximum: 6).
`LogFile`	Text			
`LogMode`	Value	0, 1	0	
`Undo`	Value	0, 1	100	

Path format of a DefaultView

When connecting via a local connection the path definition must end with the parameter `[Database Name]:`.

DefaultView=`...\DefView\[DatabaseName]\`

When connecting via a TCP/IP connection the path definition must end with the parameter `[Server Name]\[DatabaseName]:`

DefaultView=`...\DefView\[ServerName]\[DatabaseName]`

Example:

Local connection:

`D:\AleaRoot\DefView\[DatabaseName]`

TCP/IP connection:

`D:\AleaRoot\DefView\[ServerName]\[DatabaseName]`

Cluster.ini

The `Cluster.ini` file contains information about the cluster environment. It contains list of databases and nodes which can be part of OLAP Server cluster.

Parameter	Required format	Acceptable values	Default value	SOA: Stored in Config Service	Description
`[Cluster]`					
`Cluster Name`	String			Read	Dependency: System-Mode flag in `Alea.ini`
`DBs`	String	Example: DBs=Demo_DB_1:A:computer1:computer2|Genesis|	NULL		Contains a list of databases which can be started in a cluster on different nodes. If the database is configured to start in Automatic mode like <DB-Name:A>, it should contain the list of nodes on which it should start, that is, by <DB-Name:A:<Node1:Node2>>. The default mode is Manual. Dependency: System-Mode flag in `Alea.ini`
`Nodes`	String	Example: Nodes=computer1:2904|computer2	NULL		Contains a list of nodes which can be part of Cluster. Nodes also contains information about the port, but this information is not mandatory. Default port is 2904. Dependency: System-Mode flag in `Alea.ini` .

Changing the location of Alea.ini and Mdsex.ini

In some cases it is necessary to store the `Alea.ini` and the `Mdsex.ini` files in a different directory. This can be done via settings of the Windows registry.

If you change the location of `Alea.ini` and/or `Mdsex.ini` on a 64-bit operating system, you must change the registry data of the 32-bit registry (using `C:\Windows\regedit.exe`) as well as in the 64-bit registry (using `C:\Windows\SysWOW64\regedit.exe`).

Changing the location of the `Alea.ini` file for all users

1. Open the Registry Editor.
2. Navigate to the `HKEY_LOCAL_MACHINE\SOFTWARE\MIS AG\Alea` and right-click. Select **New > Expandable String Value**. Specify **Alea.ini** as the name of the new key and press the return key.
3. Right-click the new key and select **Modify**.

   In the **Value data** field, specify the path to the `Alea.ini` file. For example, `C:\Alea_INI_Keys\Alea.ini` and click **OK**.

Changing the location of the `Mdsdx.ini` file

1. Open the Registry Editor.
2. Navigate to the `HKEY_LOCAL_MACHINE\SOFTWARE\MIS AG\Alea` and right-click. Select **New > Expandable String Value**. Specify **MdsEx.ini** as the name of the new key and press the return key.
3. Right-click the new key and select **Modify**.

   In the **Value data** field, specify the path to the `MdsEx.ini` file and click **OK**.

Changing the location of the `Alea.ini` file for the current user

1. Open the Registry Editor.
2. Navigate to the key `HKEY_CURRENT_USER\SOFTWARE\MIS AG\Alea` and right-click. Select **New > Expandable String Value**. Specify **Alea.ini** as the name of the new key and press the return key.
3. Right-click the new key and select **Modify**.

   In the **Value data** field, specify the path to the `Alea.ini` file. For example `%USERPROFILE%\Local Settings\Alea\Alea.ini`. Click **OK**.

Changing the location of the `Mdsex.ini` file for the current user

To change the location of the MdsEx.ini file for the current user, repeat steps 1 to 3 and create a registry key named `MdsEx.ini` containing the path to the `MdsEx.ini` file as value.

1. Open the Registry Editor.
2. Navigate to the key `HKEY_CURRENT_USER\SOFTWARE\MIS AG\Alea` and right-click. Select **New > Expandable String Value**. Specify **MdsEx.ini** as the name of the new key and press the return key.
3. Right-click the new key and select **Modify**.

In the **Value data** field, specify the path to the `MdsEx.ini` file. and click **OK**.

Changing the location of the Config.xml File

In some cases it can be necessary to store the `Config.xml` file in a directory different from the directory `%APPDATA%\Infor\BI\OLAP`. This can be done via settings of the Windows registry.

Changing the location of the Config.xml file for all users

1. Open the Registry Editor.
2. Navigate to the key `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\MIS AG\Alea` for a 64-bit operating system or `HKEY_LOCAL_MACHINE\SOFTWARE\MIS AG\Alea` for a 32-bit operating system. Right-click and select **New > Expandable String Value**. Specify **Config.xml** as the name and press the return key.
3. Right-click the new key and select **Modify**.

   In the **Value data** field, specify the path to the `Config.xml` file and click **OK**.

Changing the location of the Config.xml file for the current user

1. Open the Registry Editor
2. Navigate to the key `HKEY_CURRENT_USER\SOFTWARE\MIS AG\Alea`. Right-click and select **New > Expandable String Value**. Specify **Alea.ini** as name and press the return key.

   Note: This is different than `HKEY_LOCAL_MACHINE`, it is the same for 64-bit- and 32-bit operating systems.
3. Right-click the new key and select **Modify**.

   In the **Value data** field, specify the path to the `Config.xml` file. For example `%USERPROFILE%\Local Settings\Alea\Config.xml`. Click **OK**.

Possible formats

There are three possibilities for formats of entered values to `HKEY_CURRENT_USER\Software\MIS AG\Alea\Config.xml` and `HKEY_LOCAL_MACHINE\Software\MIS AG\Alea\Config.xml`:

- The `Config.xml` file is located in a user-defined path.

  Example: `C:\UserDefinedPath\`
- The filename is user-defined and the file is located in a user-defined path.

  Example: `C:\UserDefinedPath\UserDefinedFilename.xml`
- The location of the xml-file is dependent on the current user.

  Example: `%USERPROFILE%\`

In this case the filename can also be changed.

Example: `%USERPROFILE%\UserDefinedFileName.xml`