



Asset Manager

Software Version: 9.61

Windows® and Linux® operating systems

Advanced Use

Document Release Date: September 2016

Software Release Date: September 2016



Hewlett Packard
Enterprise

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© 1994 - 2016 Hewlett Packard Enterprise Development LP

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: <https://softwaresupport.hpe.com/>.

This site requires that you register for an HPE Passport and to sign in. To register for an HPE Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HPE Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Support

Visit the HPE Software Support site at: <https://softwaresupport.hpe.com>.

This website provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HPE support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HPE Passport user and to sign in. Many also require a support contract. To register for an HPE Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HPE Passport login page.

To find more information about access levels, go to: <https://softwaresupport.hpe.com/web/softwaresupport/access-levels>.

HPE Software Solutions Now accesses the HPE Software Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is <http://h20230.www2.hp.com/sc/solutions/index.jsp>.

Contents

Chapter 1: Itemized Lists	16
Custom itemized lists	16
Values of custom itemized lists	17
Open custom itemized lists	17
Closed custom itemized lists	17
System itemized lists	17
Chapter 2: Historization	19
Record creation	20
Modification of a field in the table or a 1 link (for example: User of an asset)	20
Addition of a n link to another table (for example: Assets covered by a contract)	21
Deletion of a n link to another table	21
Modification of a n link to another table	22
Keeping history of features associated with records	22
Adding a feature	23
Deleting a feature	23
Modifying the value of a feature	23
Creating, deleting or modifying history lines	24
Triggering the creation of history lines	24
Chapter 3: AQL queries	25
AQL tools	25
Presentation	25
AQL	25
Queries in Asset Manager	27
Query editor	28
Overview	28
Accessing the query editor	29
Creating a query using the query editor	30
Fields used in queries	32
Writing expressions	32
Constants: Windows client only	34

Recommendations for writing AQL queries	35
Presentation of AQL joins	36
Reason for and usefulness of primary key 0 records	37
Usage of NULL	38
Self	39
CurrentUser	40
System itemized-lists	40
Hierarchic tables	41
Simplified AQL notation	42
Sorts and indexes	44
Example	44
How to force the indexes	44
Sort order	45
Precautions	46
AQL syntax	46
Conventions	47
Syntax of queries	47
Elements of a query	49
FROM clause	54
WHERE clause	55
GROUP BY clause	56
HAVING clause	57
ORDER BY clause	58
INSERT clause	58
UPDATE clause	59
DUPLICATE clause	59
DELETE clause	60
AQL function reference	60
Aggregate-type AQL functions	61
String-type AQL functions	61
Date-type AQL functions	63
Numeric-type AQL functions	65
Test type AQL functions	67
Examples of queries	67
To compare a field in the main table with a value	68

To compare a link in the main table with another link	68
To compare a link in the main table with a value	68
To compare according to a field in a table linked to the main table	69
Hierarchic tables	69
Query combining two conditions	69
Comparison of a field with numbers, dates or text	70
Query concerning a feature	70
To search records according to an expression	70
To search a field that is not populated	70
To search for the absence of a link	70
Query with alias	72
Query with variable	72
Chapter 4: SAP Crystal Reports	74
General overview	74
Why use SAP Crystal Reports (Designer)?	75
Where are the reports stored?	75
Where are the reports referenced?	75
Report types	75
Installing and configuring to enable Asset Manager clients to access reports stored in the SAP BusinessObjects Business Intelligence Platform 4.1	76
Prerequisites	76
Installing SAP BusinessObjects Business Intelligence Platform 4.1 and SAP Crystal Reports (Designer)	77
Configuration checklist for unattended and manual installation	81
Configurations to enable Asset Manager Web clients to access reports stored in the BusinessObjects Business Intelligence Platform database	86
Configuring SSO	93
Declaring the access URL to the SAP BusinessObjects Business Intelligence Platform	97
Mapping the report file name with Crystal report ID	98
Configuring the sysCoreWebCrystal calculated field	99
Migrating from BusinessObjects Enterprise XI 3.1 to BusinessObjects Business Intelligence Platform 4.1	100
Display a report	100
To display a report	101

Available reports	103
How is the report displayed	103
How does Asset Manager generate the URL address of the reports	104
Making reports available	104
Obtaining .rpt files	105
Store reports in the Asset Manager database	105
Storing reports in the SAP BusinessObjects Business Intelligence Platform	107
How to modify a Seagate Crystal report	108
Reports stored in the Asset Manager database	108
Reports stored in the SAP BusinessObjects Business Intelligence Platform	109
Creating a detail report	109
Example of utilization	109
Configuring reports under SAP Crystal Reports	109
Printing a report	111
Detail reports (Windows client only)	111
List reports and graphs	112
Identifying Crystal Reports specific to a given module	113
Associating a report with a button in a screen (Windows client) or with actions in a screen (Web client)	114
Limitations	114
Chapter 5: Dashboards	115
Creating a dashboard	115
Dashboard example	116
Chapter 6: Statistics	119
Create a statistic	119
Using a script instead of a query	121
Why use a script instead of a query?	121
Syntax of scripts	122
Optimizing performance	122
What can cause performance issues?	122
Solution to work around these performance issues	123
How do I implement this solution?	123
Zooming in on statistics details	123

Defining the maximum number of groups to represent individually	123
Making the statistics viewable	124
From dashboards	124
From business home pages	125
Displaying a statistic	125
Examples of statistics	125
Non-scripted statistics	126
Scripted statistics	131
Defining custom colors for charted statistics	139
Error handling	139
Practical Case	140
Chapter 7: Actions	141
Development best practices	141
Use a test functional domain to customize actions	142
Tag the Web services	143
Definition of an action	143
Functional domain	144
Creating an action	145
Types of actions	145
General process of creating an action	149
Populating the DDE tab	150
Populating the Messaging tab	151
Examples of actions	153
Example of an executable-type action	154
Example of a DDE-type action	154
Example of Messaging type action	154
Example of a Script type action	155
Using variables	157
Actions linked to the helpdesk	157
Actions defined by escalation schemes	157
Actions defined in the suspension slips	159
Testing an action	159
Calculate button	159
Execute button	159
Executing an action	160

Windows client	160
Web client	161
Multiple-selection in lists	161
Wizard-type actions	161
Executable type actions	162
Associating an action with a button in a screen	163
Overriding an action	163
Chapter 8: Workflow	165
Definitions	166
General overview	167
How to implement workflow	168
Using the graphical workflow editor	169
Activities	169
Events	170
Transitions	171
Other functionality	171
Example of workflow used in request approval	171
Aim	172
Prerequisites	173
Creating activities	176
Configuring events created at the same time as activities	180
Creating the start event	181
Creating transitions	181
Example of activating a workflow instance	182
Context of a workflow	185
Defining the context of a workflow instance	185
Object referenced by a workflow instance	185
Limiting workflow instances in progress for a given object	186
Workflow roles	187
Workflow role type	187
Defining the assignee of an activity	188
Workflow activities	188
User-action type activity	189
Question type activity	189
Automatic-action type activity	190

Test/ Script type activities	191
Start activity	192
Activity templates	192
Triggering activities	193
Workflow tasks	193
Creating tasks	194
Automatic action or test/ script type activity	194
Display the list of all the workflow tasks	195
Performing a user task	195
Assigning user tasks	196
Delegating a task	196
Administrating a workflow task	197
Workflow events	197
System event	198
Alarm event	198
User event	199
General conditions of activation	201
Processing of events	202
Application: Implementing a synchronous workflow scheme	206
Terminal event	207
Workflow transitions	208
Workflow alarms and time limits	208
Time limit	208
Workflow alarms	209
Workflow execution groups	210
Priority	210
Workflow tracking	211
Display the workflow instances of a record	211
Viewing information during different steps of the instance	212
Refreshing workflow schemes and instances	212
Deleting instances of finished workflows	213
Why should you delete an instance of a finished workflow?	213
Automating the deletion of terminated workflow instances	214
Technical information: Data model	218
Chapter 9: Exporting data and creating SQL views	220

Definitions	220
Export scripts	220
Export queries	221
Exporting data from the Asset Manager database	221
Exporting data using an export script	222
Exporting data using the Export the list shortcut menu item	222
Managing SQL views in the Asset Manager database	223
Recommendations	223
Defining an export script	224
Methodology	224
Defining export queries	225
Output format of an export script	227
Actions concerning SQL views	228
Executing an export script	229
Executing an export script from Asset Manager Export Tool	229
Executing an export script from DOS	230
Chapter 10: Scripts	231
Definition of a script	231
Overview	231
Information about this version of Basic	232
Data access notation	232
Applications of scripts	233
Introduction to functions	234
Definition of a function	234
Built-in functions and programmable functions	235
Function and parameter types	236
Classifying Basic functions	238
First steps in writing scripts	238
Example scenario	238
Step 1: Create the feature "Tutorial"	239
Step 2: Open the edit window	239
Step 3: Analyze and define the algorithm	239
Step 4: Draw up the Basic script	240
Step 5: Test the Basic script	240
Script libraries	241

Concepts	241
Creating a script library	242
Calling up a script in a script library	242
Tips and warnings	243
Precautions when using programmable functions	243
Format of "Date+Time" constants in scripts	244
Format of "Duration" constants in scripts	244
Read and write access to a system itemized list	245
CurrentUser virtual link	246
Old virtual link	247
Commenting a Basic script	247
Triggering an error message	248
Using recursive functions	248
Force Unicode text file	248
First example	249
Description of the problem	249
Step 1: Analyze and define the algorithm	249
Step 2: Draw up the Basic script	250
Step 3: Test the Basic script	251
Second example	251
Description of the problem	251
Step 1: Analyze and define the algorithm	251
Step 2: Draw up the Basic script	252
Step 3: Test the Basic script	252
Chapter 11: Calendars	253
Overview of calendars	253
The calendar detail	253
Using calendars	254
Impact of calendars on certain areas of functionality	254
Methodology used to create a calendar	255
Description of how to create a calendar	255
Entering general information	256
Populating the Timetable tab	256
Populating the Exceptions tab	257
Previewing the calendar	260

Chapter 12: Time zones	261
Why manage time zones?	261
Implementing time zones	262
Creating time zones	262
Managing a time zone	263
Format of Daylight time field	263
Values of the <Year> argument	264
Values of the <DaylightInfo> argument	264
Example	267
Managing time zones in Asset Manager Automated Process Manager	269
Tests to perform	269
Frequency of the test	270
Consequences for various operations	270
Creating the database	270
Connecting to a database	271
Import and Export	273
Calendars and escalation schemes	273
Chapter 13: Calculated fields	274
Definition of a calculated field	274
Usefulness of calculated fields	274
Creating a calculated field	275
Introduction	275
Methods of creation	277
Using calculated fields	280
Using a calculated field in the configuration of a list	280
Filtering the records of a table	280
Referencing a calculated field	280
Chapter 14: Wizards	282
Overview	282
Notational conventions	283
Definitions	283
Structural template	286
Wizard page template	287
General points	287

Generic structure and syntax	287
Sequencing wizards	288
Execution	289
Parameters	289
Basic functions	289
Values returned by the functions	290
Concatenating strings in Basic scripts.	290
Properties of a node	291
Declarative template	291
Defining a constant as a value for a property	292
Referencing a property	292
Defining a script as value for a property	293
Methods applicable in properties	294
Table type property	294
Using the global variables CurrentTable and CurrentSelection	295
Types of nodes	296
Root node	296
Page node	302
Transition node	304
Properties of a Transition node	305
Finish node	306
Start node	307
Timer node	308
Long and String nodes	309
Control node	309
Types of controls and associated properties	310
Common properties	311
The CHECKBOX control	314
The COMBOBOX control	314
The OPTIONBUTTONS control	315
The LISTBOX control	316
The LABEL control	321
The PROGRESSBAR control	321
The COMMANDBUTTON control	322
The DBLISTBOX control	323

The DBQUERYBOX control	326
The DBEDIT control	329
The DBTABLE control	330
The DBPATH control	330
The LINKEDIT control	330
The TEXTBOX control	332
The CHART control	332
The FILEEDIT control	335
The TICKEDIT control	335
The CALENDAR control	336
The TIMSPANEDIT control	336
The NUMBOX control	336
The COMBOEDIT control	337
The DATETIMEEDIT control	337
Using the graphical editor	338
Overview of the interface	339
Creating a new node	340
Editing the properties of a node	341
Compiling, executing and debugging a wizard	341
Example of creating a wizard	342
Example of creating a wizard	342
Step #1- Analyze your needs	342
Step #2- Define how the wizard is organized	344
Wizard programming case study	347
Frequently asked questions	354
{lbxMyListBox.Values.Count} does not work	354
{lbxMyListBox.Line(IRow)} does not work	355
{lbxMyListBox.Values.Line({lbxTmp})} does not work	355
Assigning fixed value to a property does not work	355
Executing a wizard which creates an asset in the database causes error	356
Incomplete error message appears when executing wizard	356
Performance is impacted when the "DBLISTBOX" control is used in a wizard page	356
How to allow or forbid editing certain columns in the "LISTBOX"	357

control?	
How to handle error "Component ID AcwOFFScreenFilter has already been found in the view"	357
What do I need to do to make a wizard open a detail screen?	358
What is the difference between the "COLNAME" and "COLTITLE" properties of a "LISTBOX" control?	358
The word "query" cannot be used in any wizard elements	359
Chapter 15: News	360
Definition of a news item	360
Overview of news	360
Creating a news item	361
Reading news	361
Importance of news items	361
Message to broadcast	361
News broadcast list	361
All employee groups option (SQL: bAllGroups)	362
Include sub-groups option (SQL: bChildGrps)	362
Display the news	362
Activating the news marquee	363
Data health check	364
Health check widget	364
Managing health check rules	365
Send documentation feedback	368

Chapter 1: Itemized Lists

An itemized list is a list of values proposed by Asset Manager for populating certain fields (standard fields in a detail screen, or the value of a feature), such as Title, Position, Country, Brand. They are stored in the **Itemized lists** (seltemizedList) table in the database.

This enables you to standardize the values in these fields, and facilitates data entry.

When a screen includes an itemized list field, its values appear to users in a "drop-down list". To assign a value to the field, the user simply selects an entry from the list.

List values are stored in the **Itemized list values** (amlItemVal) table in the database, and linked to their corresponding lists in the **Itemized lists** (seltemizedList) table

The out-of-the-box Asset Manager database has fields which are linked to two types of itemized lists:

- Custom itemized lists
- System itemized lists

Custom itemized lists can be **closed** (not modifiable by users) or **open** (users can add new list entries). Whether an itemized list is open or closed can be changed by any user who owns read/write access to the **Itemized lists** (seltemizedList) table.

Note: Administrators can create new fields with Asset Manager Application Designer that they can associate to either a system or custom itemized list.

This chapter includes:

Custom itemized lists	16
System itemized lists	17

Custom itemized lists

Custom itemized lists can be accessed using the **Administration/ System/ Itemized lists** link in the navigation bar.

Values of custom itemized lists

The list of "values" appearing in the detail of an itemized list contains the values that will be proposed when the user populates a field associated with the itemized list.

Values can be deleted, modified or added in the detail of the itemized list.

Open custom itemized lists

The **Type** field (SQL name: seType) field in the detail of these itemized lists is set to **Open**.

Asset Manager users can enter other values than those proposed in the list.

If a user enters a new value, it is added to the list of values in the itemized list (the list of values is shared by all users). A message requests the user to confirm the creation.

Closed custom itemized lists

The **Type** field (SQL name: seType) of the detail for these itemized lists is set to **Closed**.

Asset Manager users enter only values included in this list.

System itemized lists

The list of values in a system itemized list is defined with Asset Manager Application Designer. It can be customized by administrators.

These itemized lists cannot be edited using the **Administration/ System/ Itemized lists** link in the navigation bar.

Values in system itemized lists

The values displayed are different from the values stored in the database.

In the database, these values are stored as numbers.

Example of the **Assignment** field (seAssignment) in a portfolio item detail:

Values in system itemized lists

Value stored in the database	Value displayed
0	In use
1	In stock
2	Retired (or consumed)
3	Awaiting receipt
4	Return for maintenance
5	Return to supplier
6	Missing

There are several ways to access the values in a system itemized list:

- Using the context-sensitive help (Windows client only) for the field populated by the system itemized list.
- From Asset Manager Application Designer.
- Using the **database.txt** file describing the database structure. This file can be found in the **<Asset Manager installation folder>\ doc\ infos** directory.

Chapter 2: Historization

Any modifications made to field values and table links in the database can be tracked and recorded. Every time you create, modify or delete a value in a field for which the history is kept, Asset Manager creates a history line in the **History** tab of that screen.

In order to do this you need to specify that "history" is kept for the field or link. To do this:

1. Right-click the field or link for which you want to track changes.
2. From the shortcut menu, select **Configure object**.
3. Go to the **General** tab in the configuration screen.
4. Set the **Keep history** field to **Yes**.
5. Click **OK** to confirm.

Any modifications concerning history are saved in the database as soon as you click **OK** in the database customization window.

Note: You can also define whether history is kept for a field or a link using Asset Manager Application Designer. However, you cannot enable the history tracking functionality in the Web client.

When history is kept for a field or a link, it is available for all users of Asset Manager.

As soon as history is kept for at least one field or link, a **History** tab appears in the record detail window for this table. "History lines" are kept here, which describe in detail any modifications that have been made to a field or a link.

History lines contain several pieces of information:

- **Modified on** (SQL name: dtLastModif): Date on which the modification was carried out.
- **Author** (SQL name: Author): Person who performed the modification (login, name and first name).
- **Field or link** (SQL name: Field): Name of the field that has been modified (short description).
- **Previous value** (SQL name: PreviousVal): Previous value of the modified field (except for "comment" type fields).
- **New value** (SQL name: NewVal): New value of the modified field (except for "comment" type fields). By default, this field does not appear in the list. To display it, right-click the list then select **Utilities/ Configure list**.

Note: If you are importing a database from an older version of Asset Manager, the **New value** field will be empty for history lines.

- **Previous comment** (SQL name: memPreviousCmt): Previous value of "comment" type fields. "Comment" type fields are not handled in the same way as other fields, since they are stored differently in the database (max. size: 32767 characters).

Asset Manager behaves differently depending on the type of object that is historized.

This chapter includes:

Record creation	20
Modification of a field in the table or a 1 link (for example: User of an asset)	20
Addition of a n link to another table (for example: Assets covered by a contract)	21
Deletion of a n link to another table	21
Modification of a n link to another table	22
Keeping history of features associated with records	22

Record creation

Record creations are recorded if you have asked Asset Manager to track history of all modifications made to the identification field corresponding to the primary key of the table).

Asset Manager records:

- **Modified on:** Creation date
- **Author:** Author of the creation
- **Field or link:** "Creation"
- **Previous value:** "Creation"

Modification of a field in the table or a 1 link (for example: User of an asset)

Asset Manager records:

- **Modified on:** Date on which the modification was made.
- **Author:** Author of the modification.
- **Field or link:** Name of field modified.
- **Previous value:** Previous value of the modified field.
- **New value:** New value of the modified field.

Note: If the modified field is populated using a system itemized list, the **Previous value** and **New value** fields store the value that is displayed by the system itemized list input field rather than the value stored in the database.

For example: In the **Work orders** (amWorkOrder) table, the **Status** (seStatus) field is populated via a system itemized list. One of the entries of this itemized list is displayed as **Notified** and is stored as **0**.

The **Previous value** and **New value** fields store **Notified** and not **0**.

This behavior is valid starting with Asset Manager version 5.00.

In previous versions, the value stored in the database was used.

Addition of a n link to another table (for example: Assets covered by a contract)

Asset Manager records:

- **Modified on:** Date on which the addition was made.
- **Author:** Author of the addition.
- **Field or link:** Name of link.
- **Previous value:** References of the linked record that has been added.
- **New value:** New value of the modified link.

Deletion of a n link to another table

Asset Manager records:

- **Modified on:** Date on which the deletion was made.
- **Author:** Author of the deletion.
- **Field or link:** References of the linked record that has been deleted.
- **Previous value:** References of the linked record that has been deleted.
- **New value:** New value of the modified link (empty).

Modification of a n link to another table

Asset Manager does not record modifications made to a link. To keep a trace, you need to delete the obsolete link, then add the new one.

Keeping history of features associated with records

Note: This functionality is currently not available in the Web client.

In Asset Manager, you can keep history of features, just like for any other field in the database. Feature history concerns:

- Adding features.
- Removing features.
- Changing the value of a feature.

Several types of actions are kept in history:

Adding a feature	23
Deleting a feature	23
Modifying the value of a feature	23
Creating, deleting or modifying history lines	24
Triggering the creation of history lines	24

Adding a feature

Additions of new features are recorded if the **Keep history** (SQL name: seKeepHistory) field for the feature is set to **Yes** and if the **Keep history even when creating main record** (SQL name: bCreationHistory) option is selected.

Asset Manager records:

- **Modified on** (SQL name: dtLastModif): date when the feature was added.
- **Author** (SQL name: Author): the person who added the feature.
- **Previous value**: "Creation".
- **Field or link**: SQL name of the feature.

Deleting a feature

Deletions of features are recorded if the **Keep history** field for the feature is set to **Yes**.

Asset Manager records:

- **Modified on**: date when the feature was deleted.
- **Author**: the person who deleted the feature.
- **Field or link**: SQL name of the feature.
- **Previous value**: " Remove feature ("Value of feature")".
- **New value**: New value of feature (empty).

Modifying the value of a feature

Modifications of features are recorded if the **Keep history** field (SQL name: seKeepHistory) for the feature is set to **Yes**.

Asset Manager records:


- **Modified on:** Date when the feature was modified.
- **Author:** The person who modified the feature.
- **Field or link:** SQL name of the feature.
- **Previous value:** Previous value of the feature.
- **New value:** New value of the modified feature.

Caution: If you delete a record, all history lines are also deleted, either straightaway or via Asset Manager Automated Process Manager.

Creating, deleting or modifying history lines

It is not possible to keep history of the creation of history lines.

Triggering the creation of history lines

To trigger the creation of history lines for a feature, you must set the **Keep history** field to **Yes**. To do this, select the **Parameters** tab in the feature details and click the  button opposite the parameters line.

Asset Manager displays the screen with the details of the feature's parameters. The **Keep history** field is located in the **Constraints** tab in this screen.

When this field is set to **Yes**, Asset Manager automatically creates history lines for this feature. History lines may be viewed in the **History** tab of the table associated with this feature.

Caution: If you delete a record, all history lines are also deleted, either at the time of the deletion, or via Asset Manager Server. You cannot keep history of the creation of histories.

Chapter 3: AQL queries

This chapter includes:

AQL tools	25
Presentation	25
Query editor	28
Recommendations for writing AQL queries	35
Sorts and indexes	44
AQL syntax	46
AQL function reference	60
Examples of queries	67

AQL tools

You can perform AQL queries in the Asset Manager graphical interface, in which case they are contextual, or else perform them in Asset Manager Export;, in which case they will be non-contextual.

Presentation

This section presents the AQL language and lists the places where you might have to use queries and includes:

AQL	25
Queries in Asset Manager	27

AQL

AQL ("Advanced Query Language") is the querying language used by Asset Manager to access the Asset Manager database. It is comparable to SQL. Queries written in AQL are automatically translated into the corresponding SQL language of the database engine used.

Note: It is useful to be well acquainted with SQL and to have good background knowledge of databases before using AQL directly.

Usefulness of AQL language

AQL language is better suited to querying the Asset Manager database than SQL for the following reasons:

- Independence from the database engine

The various database engines supported by Asset Manager all use differing versions of SQL that are incompatible with each other. AQL is independent of the database engine used.

As a consequence, if you migrate to another database engine, your queries written in AQL will work the same.

For example, the **Substring** function in AQL is equivalent to **Substr** under SQL Oracle for WorkGroups and **Substring** under Microsoft SQL Server SQL.

- Generation of optimized SQL code

AQL generates SQL code optimized according to the database engine used.

This makes a big difference when using indexes. For example, when searching the full names of models by forcing the use of the indexes **ID of the model(Model_IDModelId)** and **Full name (FullName)** you would write the following in AQL:

```
SELECT FIRST_ROWS lModelId, FullName FROM amModel
```

The SQL code generated will differ according to the DBMS used and will be optimized for this. Thus, the SQL Oracle code will be:

```
SELECT /*+ FIRST_ROWS INDEX_ASC(M1 Model_IDModelId) */ M1.lModelId, M1.FullName  
FROM amModel M1
```

The Microsoft SQL Server code will be:

```
SELECT M1.lModelId, M1.FullName FROM amModel M1 ORDER BY M1.lModelId
```

- Simplified access to the Asset Manager database

AQL simplifies the handling of links and joins. This greatly facilitates access to the database when writing queries as compared to using SQL directly.

In addition, AQL simplifies access to features, allowing you to use them as direct fields in their related tables.

AQL also facilitates the utilization of calculated fields.

Specifics of AQL compared to SQL

AQL does not support DDL ("Data Definition Language") orders.

AQL comprises extensions that simplify the handling of joins, features and calculated fields.

Caution: You should never write to the Asset Manager database directly using SQL statements.

Queries in Asset Manager

Queries enable you to combine criteria concerning information in a table or in linked tables.

You can use queries to:

- Create filters on record lists. In this case, the queries are, in general, simple and based on the "Where" clause.
- Define views.
- Define export conditions in the export module.
- Create SAP Crystal Reports.
- Create wizards.
- When you use the Asset Manager API.
- If Asset Manager is used as a DDE server.

AQL ("Advanced Query Language") is Asset Manager's internal querying language intended to simply access to the Asset Manager database.

Asset Manager includes an editor that enables you to draw up queries in AQL:

- Either indirectly by using the graphic interface.
- Or by writing the query directly in AQL.

In order to be illustrative of the capabilities of AQL, the examples given later use all of the AQL syntax. The SELECT, WHERE, and FROM clauses, in particular, are explicitly shown. Certain functions, such as query filters (where the user only defines the WHERE clause in the AQL query) or the expression builder, greatly simplify the creation of queries for the user (certain clauses are not visible). These examples cannot be used for these functions.

Query editor

Asset Manager includes a query editor. This tool enables you to design queries and preview their results. It is particularly aimed at database administrators and power users.

Note: Query editor cannot be used in the wizards.

Note: The preview function is not applicable to the Asset Manager Web client and some simplified versions of query editor in Asset Manager Windows client.

This section includes:

Overview	28
Accessing the query editor	29
Creating a query using the query editor	30
Fields used in queries	32
Writing expressions	32
Constants: Windows client only	34

Overview

The query editor makes it possible to design queries:

- Either by using the graphic interface (assisted query design).
- Or by writing the query directly in AQL.

If you are using the Asset Manager Windows client, you get to see a real-time transcription of your query in SQL using the Preview function, whether you use the graphic method or write your query directly in AQL (both approaches are frequently combined). However, you cannot write your queries directly in SQL.

Query editor - composition modes



Using the query editor, a power user or an administrator can create, modify or delete AQL queries. These queries can be used in the appropriate context by their author or other users.

Accessing the query editor

- Windows client

You can access query editor in the Asset Manager Windows client:

- Via the **Tools/ Queries** menu item. Using this menu, you can create queries for your own use, which can also be used freely by other users. The queries can be executed:

Note: The Query tab has two tabs, **Filter (WHERE clause)** and **Preview**:

- The **Filter (WHERE clause)** tab is a graphical interface that determines the conditions of your query. It defines the elements of the WHERE clause.
 - The **Preview** tab displays the transcription of your query in SQL code and enables you to test it.
- Via the query filter when the main table of the query is displayed.
 - Via the numerous functions of Asset Manager that call on queries: Access restrictions, query filters, list configuration, and so on.
 - Via external programs: Asset Manager Export Tool, and so on.

The version of the query editor that is shown is simplified to a certain degree according to the context.

Example: Let's suppose that you have a query such as the following:

```
SELECT [FIRST_ROWS] <field>[, <field>...] FROM <table> [WHERE <clause>] [ORDER BY <clause>]
```

In the simplified versions of the query editor (simple filters, query filters, and so on), you only have to define the WHERE clause of the query. The other components of the query (starting table, fields, and so on) are implicit. For example, in the case of a query filter, the table is that on which the filter is applied, the fields and the sort conditions are the columns and the sort conditions which are defined via the **Utilities/ Configure list** shortcut menu item. The same is true for the query editor accessed via the **Tools/ Queries** menu item.

Thus, the following query given in full:

```
SELECT self FROM amModel WHERE Brand.Name='Compaq'
```

is written as follows when used in a query filter (only the WHERE clause is explicitly given) used on the table of models:

```
Brand.Name= ' Compaq '
```

On the other hand, the **Configure list** command enables you to access a more comprehensive version of the query editor:

- The **Columns and sort** tab defines the fields to be displayed in columns and the sort conditions (these sort conditions correspond to the ORDER BY clause).
- The **Force indexes** box replaces the FIRST_ROWS clause in the SQL code.
- The **Filter (WHERE clause)** tab defines the "WHERE" clause.
- The table is implicit.
- Web client

You can only access query editor via the Queries table.

- Select the **Administration/ Queries** menu.
- Click the **New** button.
- Select the **Query** tab.
- Click the icon above the text box, the query editor displays in a pop-up window.

Creating a query using the query editor

Step 1: Populate the fields at the top of the query detail.

You must specify the starting table of your query.

If you want the query you are creating to be accessible to other users, uncheck the **Not shared** option (SQL name: bPrivate) on the **General** tab.

Note: The administrator has access to all queries stored in the database, even those queries that are **Not shared**.

Once you have filled in the basic information on the query, click the **Query** tab to work on the query.


Step 2: Define the filter conditions


The Asset Manager query editor enables you to define conditions that combine fields, calculation expressions, constants and operators.

You can define one or more filter conditions.

- Windows client

To define a filter condition in the Windows client:

- a. From the starting table, select a field, constant or expression (**Field 1**), that you will compare with a field, constant or expression (**Field 2**).
- b. Confirm the filter condition by transferring it to the lower part of the screen using the  button.
- c. To define several filter conditions linked by logical operators such as **AND** and **OR**, construct the other filter conditions and confirm them using the **AND** or **OR** buttons.
- d. Confirm the query by clicking **Modify** in the query detail.

Note: To modify the filter conditions, click the  button to clear the contents of the window, or modify the AQL code directly.

Note: In place of the graphic tool, you can enter your query directly in AQL in the text box at the bottom of the **Filter (WHERE clause)** tab.

- Web client

To define a filter condition in the Web client, select fields, functions and operators from the respective tabs.


The Web client contains the following types of operators, which enable you to use the same interface to build expressions and define filter conditions

- Arithmetical
- Relational
- Logical

Note: The Asset Manager Web client has different elements (for example, fields, operators) of a query defined in one text field, as opposed to in distinct fields. Make sure to select and arrange the elements involved in the query in the correct sequence.

Step 3: Preview the execution of the query (Windows client only)

To test the query and see its transcription in SQL language:

1. Go to the **Preview** tab in the query detail.
2. Click the  icon: Asset Manager shows a preview of the results of the query, in the form of a list of records. The number of records satisfying the query is shown at the bottom right of the window.


Note: The SQL code contained in the **Preview** tab cannot be modified directly.

Fields used in queries

When defining filter conditions in queries, you can call on:

- A field in the table concerned by the query.
- A linked field.
- Features associated with the table.

Writing expressions


Expressions  enable you to perform calculation operations in your query. For example, you can use the `Count` function to count the number of resultant records of a query.

To write an expression, you can either:

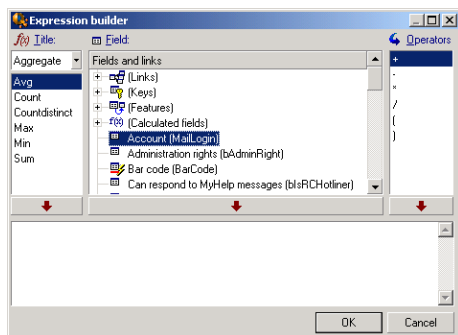
- Enter it directly in the corresponding field.
- Or use the Asset Manager expression builder.

Windows client


In the Windows client, the expression builder is integrated with, and only accessible within query editor.

To access the expression builder, from the **Filter (WHERE clause)** tab of the query editor, click the  button adjacent to the box for the value of a field.


This button is only available if the type of the field (Field 1 or Field 2) is populated with **Expression**.



The expression builder comprises three columns:

- The **Function** column lists existing AQL functions. Clicking  applies a filter on the list of AQL functions according to their type: "Aggregate", "String", "Date", "Numeric", "Test".
- The **Field** column lists the different fields that can be used in a query.
- The **Operators** column lists the operators that can be used in the expression.

To insert a **Function**, **Field** or **Operator** in the expression:

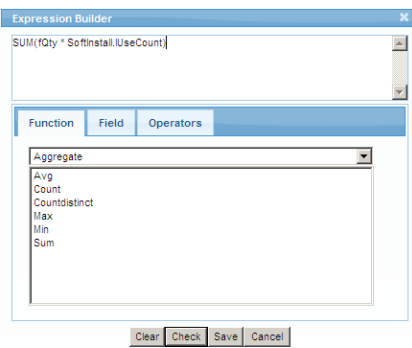
1. Select the function, field or operator.
2. Click .

Once the expression is defined, click **OK** to transfer it over to the **Filter (WHERE clause)** tab in the query detail.

Web client

In the Web client, the expression builder is a stand-alone tool and independent of query editor. But both tools have very similar interface, functions, and usage. The only difference is that expression builder has few operators than query editor.

To access the expression builder, click the icon above the text box for expression edit, the expression builder displays in a pop-up window.



The expression builder comprises three tabs:

- The **Function** tab contains various functions grouped by categories. The list of functions available will change if a different category is selected from the dropdown list. Double-click a function to insert it in the edit box at the top of the expression builder.
- The **Field** tab is contextual and contains the fields and links available for the current Asset Manager database table. Click a field or link to insert it in the edit box.
- The **Operators** tab contains the arithmetical operators applicable to an expression. Double-click an operator to insert it in the edit box.


The buttons at the bottom of expression builder facilitate the expression construction:


- **Clear** clears the content of the edit box without warning.
- **Check** checks the syntax of the expression. A message is displayed on top of the edit box to indicate whether the expression is valid or not.
- **Save** first validates the syntax of the expression. If the syntax is valid, the window is closed and the expression is saved in the Asset Manager database. Otherwise, the window remains open and an error message is displayed in the window.
- **Cancel** closes the window without any modification.

Constants: Windows client only

Constants *x* are fixed values that you assign to selection criteria. For example, if you are searching all models with brand **3Com**, you assign the constant value **3Com** to the **Brand.Name** linked field in the table of models.

To select constant:

1. Click the  icon.
2. A selection window is displayed that shows the values present in the database for the field specified as search condition.

Note: Even in the case of "Itemized list" type fields, the window displayed by clicking the  icon only shows those values used in the database.

Recommendations for writing AQL queries

We recommend reading this section before starting to write queries in AQL.

This section deals with:

- Notation specific to AQL.
- The specificities of AQL and the Asset Manager database that have an effect on the optimal design of queries.

The sections entitled [AQL syntax](#) and [AQL function reference](#) complement this section.

Caution: Queries written in AQL use the SQL names ("SQLName") of fields, links and table in the database. Refer to the **Database.txt** and **Database.xml** file that describes the structure of the database and which includes an exhaustive list of these names.

This file is located in the following folder: **[Asset Manager installation folder]/doc/infos**. The documentation must be installed to have access to this file.

This section includes:

Presentation of AQL joins	36
Reason for and usefulness of primary key 0 records	37
Usage of NULL	38
Self	39
CurrentUser	40
System itemized-lists	40
Hierarchic tables	41
Simplified AQL notation	42

Presentation of AQL joins

Definition

A join consolidates multiple data tables in a single query.

AQL joins

Asset Manager's database description, besides defining the tables and fields, defines the links between tables. This enables you to automate the generation of joins at AQL level.

AQL links are expressed as:

```
Link[.Link[.Field]]
```

By simplifying the joins in such a manner, AQL also simplifies the creation of the majority of queries used for the Asset Manager database.

Example

The following query, written in AQL, returns for each model:

- Its **ID (IModelId)**.
- Its **Full name (FullName)**.
- The **Name** of the table linked to the brands (**amBrand**).

```
SELECT IModelId, FullName, Brand.Name FROM amModel
```

Here is the same query written in SQL Oracle or Microsoft SQL Server:

```
SELECT M1.IModelId, M1.FullName, B2.Name FROM amModel M1, amBrand B2 WHERE  
M1.IBrandId=B2.IBrandId
```

The two joins between the **Models** table (**amModel**) and the **Brands** table (**amBrand**) are handled automatically in AQL. Using the Asset Manager's graphic query editor, you just have to click the fields of a selected table or linked table in a tree-structured list to create the corresponding AQL code.

Location of this file: See **Asset Manager - Installation and upgrade** guide, chapter **.ini and .cfg files**.

Reason for and usefulness of primary key 0 records

Primary key "0" records

The Asset Manager data module has certain specificities:

- The primary and foreign keys of each table are "32-bit integer" type fields.
- A foreign key that does not point to a record is set to "0" (and not "NULL").
- Each table has an empty record whose primary key is set to "0".

Usefulness

With these primary key "0" records, the results of a query using a non outer join between two given tables, A and B, can include records from table A which are not linked to any real record in table B (link not populated). These are records in table A, which are linked to the primary key "0" record in table B.

Example:

The following query, written in AQL, returns for each portfolio item's asset tag, the name of its user and its supervisor:

```
SELECT AssetTag, User.Name, Supervisor.Name FROM amPortfolio
```

A portfolio item that is not assigned to a user and/or supervisor appears in the results of the query. At database level, such an asset is linked to the primary key "0" record in the Employees and departments table.

Reason for these specificities

This section explains why primary key "0" records are used, whereas a query using an external SQL join can select records in table A that are not linked to a record in table B.

Primary key "0" records enable you to compensate for the fact that certain RDBMs do not handle multiple outer-joins: Using primary key "0" records, SQL queries generated from AQL queries do not use outer joins.

Example:

The AQL query below searches for each portfolio item, its asset tag and name of location of its user. The results will include assets that do not have a user and assets whose users do not have a location.

```
SELECT AssetTag, user.location.name FROM amPortfolio
```

If the generated SQL code used the outer joins of the DBMS, the SQL code generated for certain database engines would be:

```
SELECT a.AssetTag, l.name FROM amPortfolio a, amEmplDept e, amLocation l WHERE  
a.lUserId *= e.lEmplDeptId AND e.lLocaId *= l.lLocaId
```

This code is not supported by certain database engines since it uses several outer joins one after another.

However, since there is a primary key "0" record in the table of departments and employees and locations, it is not necessary to use outer joins. Asset Manager thus generates SQL code without outer joins:

```
SELECT l.name FROM amPortfolio a, amEmplDept e, amLocation l WHERE a.lUserId =  
e.lEmplDeptId AND e.lLocaId = l.lLocaId
```

This query gives the expected results, since the **User** and **Location** links still point to a record in the table of departments and employees or locations (they point to the primary key "0" record if the link is not populated).

Consequences

- It is important to take these records into account in the queries that you write, especially when using aggregate functions.

Example:

```
SELECT count(AssetTag) FROM amPortfolio
```

If you execute the above query, which counts the number of assets in the table of assets, the primary key "0" record is taken into account in the results. You therefore need to decrease the results by 1 to obtain the real number of assets in the database.

- It is rarely necessary to have to generate outer joins at DBMS level.

Note: Note: If you really want to generate outer joins at the DBMS level, use the "=* and "=*" AQL operators.

Usage of NULL

Asset Manager uses the NULL value of the DBMS in two cases only:

- For an empty "Text" type field.
- For a non populated "Date" or "Date and time" type field.

AQL allows you to use several equivalent syntaxes, as shown below. It converts them to the equivalent valid SQL code of the database engine.

For empty "Text" type fields, you can use any of the following syntaxes, since the NULL value will be stored in the database:

WHERE <text field> = NULL

WHERE <text field> IS NULL

WHERE <text field > = "

For non-populated "Date" or "Date and time" type fields, you can use any of the following syntaxes, since the NULL value will be stored in the database:

WHERE <field date or date+hour> = NULL

WHERE <field date or date+hour> IS NULL

WHERE <field date or date+hour> = []

Note: Note: When a numeric field is not populated, its value is "0". Similarly, the absence of a link is described as "Link = 0" or "foreign key = 0". Example: "Location=0" or "LocalId=0".

Self

"Self" is an expression that is equivalent to the description string on the table to which it is applied.

Using "Self" enables you to simplify queries and take any customization of the Asset Manager database into account.

Example:

If the description string of the table of departments and employees is:

```
[Name], [FirstName], ([Phone])
```

The AQL query:

```
SELECT self FROM amEmp1Dept
```

Is equivalent to:

```
SELECT (((((Name + ',') + FirstName) + '(') + Phone) + ')') FROM amEmp1Dept
```

CurrentUser

"CurrentUser" enables you to write queries that depend on the person connected to the database.

"CurrentUser" can be used as an expression, for example in a query, or as a link. You have to enter this expression as it is not offered by the query editor.

Used as "expression"

Example: We are looking for all the portfolio items used by the employee connected to the database.

```
SELECT lPortfolioItemId FROM amPortfolio WHERE User = CurrentUser
```

Used as "link"

"CurrentUser" can be considered as a link that is found in every table and that points to the record corresponding to the current user in the table of departments and employees.

- With the form "CurrentUser", this function points to the record corresponding to the current user.
- With the form "CurrentUser.Field", this function returns the value of the field for the current user.

Example: When an action is triggered by the connected user, it is possible to contextually trigger another messaging type action, which automatically sends a warning message to the connected user. You only need to populate the detail of the action.

System itemized-lists

If an AQL query uses a system itemized list, you must use the values that are stored in the database and not those which are displayed on screen.

Example:

The following query selects those contracts whose **Type** field (SQL name: seType) is set to **Master lease**:

```
SELECT Self FROM amContract WHERE seType = 1
```

The **Type** field (SQL name: seType) is a system itemized list. The values stored in the database are:

- 0 for **Other**
- 1 for **Master lease**
- 2 or **Lease schedule**
- 3 for **Insurance**
- 4 for **Maintenance**

Note: The values of system itemized list can be found via Asset Manager Application Designer, or the **database.txt** file, which describes the structure of the database.

This file is located in the following folder: **[Asset Manager installation folder]/doc/infos**

Hierarchic tables

All the hierarchic tables contain:

- A "FullName" field.
- A "sLvl" field.

"FullName" fields

For each record in a hierarchic table, the "FullName" field stores the value of a field of the record, preceded by a tree structure constituted by the values of the fields of parent records until root level.

Values are separated by the "/" character without spaces. This character also appears at the start and at the end of the tree-structure.

Examples:

- For the table of assets, the "FullName" field stores the Asset Tag of the asset preceded by the Asset Tag of its parent asset, that in turn preceded by the Asset Tag of its parent asset, and so on.

```
FullName = '/PC118/DD054/CR012/'
```

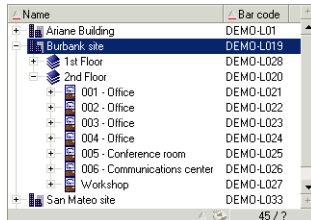
- In the table of locations, the "FullName" field stores the name of the location preceded by the names of parent locations.

```
FullName = '/Milwaukee/Water St. Site/Building A/5th floor/'
```

"sLvl" fields

For each record in a hierarchic table, the "sLvl" indicates its level in the tree-structure.

Root level is level 0.



Name	Bar code
Arise Building	DEMO-L01
+ Busbank site	DEMO-L019
+ 1st Floor	DEMO-L028
+ 2nd Floor	DEMO-L020
+ 001 - Office	DEMO-L021
+ 002 - Office	DEMO-L022
+ 003 - Office	DEMO-L023
+ 004 - Office	DEMO-L024
+ 005 - Conference room	DEMO-L025
+ 006 - Communications center	DEMO-L026
+ Workshop	DEMO-L027
+ San Mateo site	DEMO-L033

The following query selects the "Sales Head Office" record and its sub-components:

```
SELECT Self FROM amEmplDept WHERE (FullName LIKE '/Sales Head Office/Sales/%') AND (sLvl >= 1)
```

The following query selects the "Sales Head Office" record but not its sub-components:

```
SELECT Self FROM amEmplDept WHERE (FullName LIKE '/Sales Head Office/Sales/%') AND (sLvl = 1)
```

The following query selects the sub-components of the "Sales Head Office" record but not the "Sales Head Office" record itself:

```
SELECT Self FROM amEmplDept WHERE (FullName LIKE '/Sales Head Office/Sales/%') AND (sLvl > 1)
```

Simplified AQL notation

This section lists the notation that can be used to simplify the writing of AQL queries:

Foreign keys

In clauses other than the SELECT and ORDER BY clauses, the SQL name of a link that is not followed by a period is equivalent to the SQL name of the associated foreign key.

Example: the clause:

```
WHERE location = 0
```

Is equivalent to:

```
WHERE lLocaId = 0
```

Where "location" is the SQL name of the "Location" link from the table of departments and employees to the table of locations, and "lLocaId" is the SQL name of the associated foreign key in the table of assets.

Description strings

In SELECT and ORDER By clauses, the SQL name of a link that is not followed by a period is equivalent to the join <SQL name of link >.self, which is in turn equivalent to <SQL name of link>.<Description string>.

Example:

If the description string of the table of departments and employees is:

```
[Name], [FirstName] ([Phone])
```

Then the AQL query:

```
SELECT user FROM amPortfolio
```

Is equivalent to the query:

```
SELECT user.self FROM amPortfolio
```

That is itself equivalent to:

```
SELECT (((((User.Name + ',') + User.FirstName) + '(') + User.Phone) + ')') FROM  
amPortfolio
```

Features

AQL gives you direct access to the features of a table, as if they were direct fields in the table. To search feature values for a given table, you just need to write the SQL name of the feature (**fv_** prefix).

Example: The following query searches the values of the feature with SQL name **fv_WorkUnit** for the **Departments and Employees** table (**amEmpDept**):

```
SELECT fv_WorkUnit FROM amEmpDept
```

Calculated fields

AQL facilitates the use of calculated fields associated with a table.

You just need to write the SQL name of the calculated field (**cf_** prefix).

Sorts and indexes

AQL allows two strategies for queries that use sorts (ORDER BY clause):

- A mode whereby Asset Manager forces the indexes indicated in the query, when used, and displays the results as and when they are retrieved.
- A mode whereby Asset Manager does not force the indexes indicated in the query. In this case, the DBMS determines how the data is sorted.

This section includes:

Example	44
How to force the indexes	44
Sort order	45
Precautions	46

Example

In the case of the following query:

```
SELECT IModelId, Brand FROM amModel ORDER BY Brand
```

- Access without **Forcing the indexes**: the database engine scans the table in full without using the "Brand" index indicated in the query. It searches all data items satisfying the query, sorts them according to the "Brand" and sends them to the user. The results will only be displayed after a certain period of time.
- In the other case: The database engine uses the "Brand" index, and displays the results as and when they are found. The first data items are thus shown rapidly, but the overall processing time is longer.

How to force the indexes

The way you force the utilization of the indexes depends on the way that you create the query.

Via the Configure list menu

You can configure the data-access type for each list in Asset Manager. This is the case for both main lists and list contained in tabs. To do this:

1. Go to the list you want to configure.
2. Right-click.
3. Select **Utilities/ Configure list** from the shortcut menu.
4. In the **Columns and sort** tab, check the **Force indexes** box to use the indexes indicated in the query and display the results as and when they are generated. Uncheck this box to select the other type of access.

When using Asset Manager with Oracle, **Force indexes** is controlled by the **OraFirstRowsHint** parameter, which is stored in the database parameters section of the amdb.ini file. The **OraFirstRowsHint** parameter adds a SQL hint to the SQL query used to display the list. By default, its value is **FIRST_ROWS(100)** when the **Force indexes** check box is selected. To override the default setting, you can set it to another value, like `OraFirstRowsHint=FIRST_ROWS(1)`.

Note: Modifying the **OraFirstRowsHint** parameter may have performance impact on your queries. Verify it in your test environment first.

Using AQL

If you write a query directly in AQL, you can force the use of indexes via the "FIRST_ROWS" clause.

Example:

```
SELECT FIRST_ROWS AssetTag FROM amAsset ORDER BY AssetTag
```

Note: If the sort focuses on the system itemized lists (for example, on the Features table in the **seDataType** field), there is a possibility that the sort is not valid when an index is forced.

Sort order

The sort order depends on:

- The database engine.
- The use of indexes.

Under Microsoft SQL Server

The sort order depends on a parameter set when the database is created. These engines can be configured to be case sensitive or to take accented characters into account, and so on.

Precautions

With complex queries it often difficult to determine whether it is better in terms of performance to force the indexes or not. In practice, we recommend performing tests before making a final decision.

In particular, we recommend testing with and without the indexes forced in the case of a list that is filtered, be it explicitly (via a simple filter, query) or implicitly (via access restrictions).

AQL syntax

Using AQL requires familiarity with SQL language. However, a detailed description of the syntax of SQL is beyond the scope of this manual. For further information, consult the appropriate reference documentation.

This section includes:

Conventions	47
Syntax of queries	47
Elements of a query	49
FROM clause	54
WHERE clause	55
GROUP BY clause	56
HAVING clause	57
ORDER BY clause	58
INSERT clause	58
UPDATE clause	59
DUPLICATE clause	59
DELETE clause	60

Conventions

Here are the conventions used to describe the syntax of AQL:

AQL - syntax conventions

[]	These brackets surround optional items. Do not type in these brackets.
< >	These brackets surround logical items. Do not type in these brackets.
	The vertical bar indicates that choices are mutually exclusive.
...	The ellipsis indicates that the preceding text may be repeated once or several times.
FROM	Terms in uppercase letters indicate literal expressions.

Syntax of queries

Simple queries

```
SELECT [DISTINCT] [FIRST_ROWS] <Selection list>
```

```
[FROM clause]
```

```
[WHERE clause]
```

```
[GROUP BY clause]
```

```
[HAVING clause]
```

```
[ORDER BY clause]
```

Sub queries

AQL supports the use of sub-queries in the place of fields.

Note: In sub-queries, the SELECT statement only authorizes a single expression.

```
(SELECT [DISTINCT] <expression>
```

```
[FROM clause]
```

```
[WHERE clause]
```

[GROUP BY clause]

[HAVING clause]

)

Caution: Sub-queries must be contained with parentheses.

Example of utilization:

```
SELECT Self FROM amAsset WHERE mPrice >= (SELECT Max(mPrice)/2 FROM amAsset)
```

UNION type queries

UNION enables you to group together the results of several queries:

```
SELECT <selection list>
```

[FROM clause]

[WHERE clause]

[GROUP BY clause]

[HAVING clause]

[UNION | UNION ALL | INTERSECTS | MINUS]

```
SELECT <selection list>
```

[FROM clause]

[WHERE clause]

[WHERE clause]

[GROUP BY clause]

[HAVING clause]...

[ORDER BY clause]

Elements of a query

Fields and links

Queries involve fields and links in the Asset Manager database.

You can indicate the name of a field:

- In reference to the starting table of the query. In this case, it is not necessary to specify the name of this table:

[Link. ...[Link.]]<field>

Examples from the **Portfolio items** table (**AmPortfolio**):

Model
User.Name
User.Location.Name

- As an absolute reference. In this case, you need to indicate the name of the table to which the field belongs:
 - Either by declaring the table in the `FROM` clause and using its name (or a possible alias):
 - <table.[link...]field>**
 - <alias.[link...]field>**
 - Or by not declaring the table in the `FROM` clause and instead using ":":
 - <table.[link...]field>**
 - <table[_alias]:[link[_alias]...]field>**

These last two notations are particularly useful if you cannot use the `FROM` clause.

For example, when writing a query in Asset Manager, you only have access to the `WHERE` clause. The starting table of the query is implicit (table on which a filter is applied, **Table** field (**TableName**) in query detail, and so on). However, you may need to use other tables in the query. In this case, the ":" notation allows you accomplish this.

Constants

The following syntax is valid for the constants that may be involved in queries.

- Numeric constants

The period is used as decimal separator.

Examples:

12

52.23

- Text type constants

They are contained within single quotes.

Examples:

'Computer'

'Monitor'

Note: Use the escape sequence `"` to represent literal single quotes in the text constant. For example:

```
'a city with a ''' in its name'
```

- Date or time type constants;

Date or time type constants are contained between `#` characters. Their format must respect the following rules:

- Years are expressed on 4 figures.
- Dates are expressed as Year-Month-Day.
- Time is expressed as Hours-Minutes-Seconds.
- The 24 hour clock is used (and not the 12 hour clock with A.M. or P.M.).
- The separator used in dates is the `"/"` or `"-"` character.
- The separator used for time is the `":"` character.
- Months, days, hours, minutes, and seconds; these are expressed as 2 figures.
- When date and time are expressed together, the data always precedes the time and they are separated by a space.

Examples:

```
#yyyy-mm-dd hh:mm:ss#
```

```
#yyyy-mm-dd#
```

#hh:mm:ss#

#2004-01-01 01:00:03#

Variables

Instead of specifying a constant (see [Constants](#)), you can insert a variable whose value is populated by the user when the user selects a filter that uses this query.

Different syntax can be used to define variables:

- \$Variable

Using the following syntax:

```
<SQL name of the field or link> = $<Variable name>
```

The query filter user is requested to populate the **<Variable name>** field before executing the request.

<Variable name> may not contain any spaces.

- amDbVal()

Using the following syntax:

```
amDbVal(<SQL name of the field or link>,'<Label>','<Default value>')
```

The query filter user is requested to populate the **<Label>** field before executing the request.

The field to populate displays **<Default value>** by default. **<Default value>** must use the syntax described in section [Constants](#).

If **<SQL name of the field or link>** designates a link, **<Default value>** must include the primary key of the linked record.

If **<SQL name of the field or link>** designates a system itemized list, **<Default value>** must contain the number stored in the database.

Expressions

Expressions are formed using:

- Constants
- Fields

- Functions
- Sub-queries

You can combine these elements with operators and parentheses to build complex expressions.

Comparison expressions take the form:

<expression> <comparison operator> <expression>

Logical expressions take the form:

<comparison operator> <AND | OR> <comparison operator>

You can use parentheses to combine several logical expressions.

Operators

- Logical operators

Logical operators are applied to link two expressions:

AQL - logical operators

Operator	Meaning
AND	Logical "AND"
OR	Logical "OR"

In order to optimize a query, it is sometime wise to avoid logical operators if a comparison operator can be used instead. The following example illustrates how to optimize a query filter used to select portfolio items whose **Assignment** field (SQL name: seAssignment) is set to **Awaiting delivery** or **Return for maintenance**. The values of these two elements of a system itemized list are "3" and "4" respectively. It is therefore possible to write:

```
(seAssignment=3) OR (seAssignment =4)
```

The last value of the system itemized list being "4", it is preferable to write the query as follows:

```
seAssignment >=3
```

- Comparison operators

Comparison operators are used to compare two expressions.

AQL - comparison operators

Operator	Meaning
=	Equals
<> =!	Different than
>	Greater than
<	Less than
>=	Greater or equal to
=<	Less than or equal to
=*	Right outer join. Because of the way that AQL handles links, the use of this operator is limited
*=	Left outer join. Because of the way that AQL handles links, the use of this operator is limited.
LIKE NOT LIKE	Works like the = operator and allows you also to use "wildcard" characters. The following "wildcard" characters are available: "% " replaces any character string. " _ " replaces any single character. Depending on the database engine used (SQL Server supports it, Oracle for WorkGroups doesn't): [abc?] lets you define a list of possible characters (no space between the possible values.) [a-c] lets you define a range of possible values.
IS NULL	Tests whether the value of a field is "NULL" or not.
IS NOT NULL	Asset Manager only authorizes the "NULL" value for empty text fields and for Date or Date+Time fields that are not populated.

- Operators specific to sub-queries

You can compare a value to the results of a sub-query using the following operators:

- = **ANY (sub-query)**
- = **ALL (sub-query)**
- = **SOME (sub-query)**

Example:

- The following Asset Manager Export query provides the list of portfolio items whose brand is used at the Milwaukee site. (This query does not include sub-locations):

```
SELECTDISTINCT lModelId, Model.Brand FROM amPortfolio WHERE Model.Brand = ANY  
(SELECT Model.lBrandID FROM amPortfolio WHERE Location.FullName LIKE '/Milwaukee  
site')
```

For this script, following a linked record implies using the foreign key of the record.

Selection list

Selection lists define the items to be extracted or displayed. They specify the `SELECT` statements in queries.

A selection list is made up of one or more expressions separated by commas:

<expression> [, <expression>...]

Each expression can be linked to an alias. For example:

```
SELECT MrMrs, (Name + FirstName) Identity FROM amEmplDept
```

This is particularly useful at the level of export queries to attribute a name to the exported columns.

Note: Certain DBMSs limit the number of expressions in a given `SELECT` statement.

FROM clause

The `FROM` clause indicates the table or tables concerned by a `SELECT` statement.

AQL authorizes the utilization of aliases for table names.

Syntax

```
FROM <table name> [table alias] [, <table name> [table alias] ... ]
```

Starting table of a query

The first table indicated in the `FROM` clause of a query is the starting table of the query.

If a query uses a field whose table is not specified, AQL considers that it belongs to the starting table of the query. The AQL `FROM` clause differs from the clause with the same name in SQL.

For example, in the following sub-query, AQL searches the **AssetTag** field in the **amAsset** table:

```
SELECT AssetTag FROM amAsset
```

Number of tables in a query

The number of tables authorized in a query depends on the DBMS used.

Example:

- Oracle: You can use as many tables as you like.
- Microsoft SQL Server : You are limited to 16 tables in a query.

Caution: When counting the number of tables in a query, do not forget to take into account those tables that are not explicitly mentioned, in particular if the query uses links. Also look out for the "fv_" notation (search of feature values) which generates an additional join at DBMS level. Similarly, the "cf_" notion (calculated fields) can generate additional joins.

Examples

```
FROM amPortfolio  
FROM amPortfolio a, amLocation l
```

The following queries are equivalent:

```
SELECT AssetTag FROM amAsset  
SELECT a.AssetTag FROM amAsset a  
SELECT amAsset.AssetTag FROM AmAsset
```

WHERE clause

The AQL `WHERE` clause is equivalent to the clause with the same name in SQL.

It specifies the search conditions, specifying the elements to extract from the database. These conditions can also express themselves in `HAVING` clauses.

Syntax

```
WHERE <search conditions>
```

Writing search conditions

In the majority of cases, you will need to write the conditions using the following form:

```

<WHERE | HAVING> [NOT] <expression> <comparison operator> <expression>
<WHERE | HAVING> [NOT] <logical expression>
<WHERE | HAVING> [NOT] <field> [NOT] LIKE 'xxxxx'
<WHERE | HAVING> [NOT] <logical expression> <AND | OR> <logical expression>
<WHERE | HAVING> [NOT] <field> IS [NOT] NULL

```

In some other case, you may need to write more complex queries, such as:

```

<WHERE | HAVING> [NOT] EXISTS (<sub-query>)
<WHERE | HAVING> [NOT] <expression> [NOT] IN (<list of values> | <sub-query>)
<WHERE | HAVING> [NOT] <expression> <comparison operator> <ANY | ALL> (<sub-query>)

```

GROUP BY clause

The AQL `GROUP BY` clause is equivalent to the clause with the same name in SQL.

Syntax

```
GROUP BY <expression without aggregates> [, <expression without aggregates>]...
```

Writing tips

`GROUP BY` specifies subsets of the table. The subsets are defined in the `GROUP BY` clause by an expression, which can be the name of a field, for example.

If aggregate functions are included in the selection list of the `SELECT` statement, `GROUP BY` searches the resulting value for each subset. These resultant values can be used in a `HAVING` clause.

When a query makes use of the `GROUP BY` clause, each expression of the selection list must provide a single value for each subset.

GROUP BY - Examples

The following query gives the total number of brands present in the database. For each asset with an associated brand, Asset Manager returns an instance on the brand.

```
SELECT Count(Model.Brand.Name) FROM amAsset
```

By using the `GROUP BY` clause, we obtain a list of brands and the number of assets of each brand:

```
SELECT Model.Brand.Name, count(IAsId) FROM amAsset GROUP BY Model.Brand
```


HAVING clause

The AQL `HAVING` clause is equivalent to the clause with the same name in SQL.

Syntax

`HAVING <Search conditions>`

Differences with the `WHERE` clause

It specifies the search conditions like the `WHERE` clause. However, these two clauses differ as follows:

- The `HAVING` clause specifies the restrictions to be applied to aggregate functions in the selection list. The restrictions affect the number of resultant lines but do not affect the calculations linked to aggregate functions.
- When the query uses an `WHERE` clause, the search conditions restrict the lines subject to aggregate calculation functions but do not affect the resultant lines.

Examples

Example of query where the `WHERE` clause is equivalent to the `HAVING` clause:

The following query returns the list of brands whose name starts with a letter after the letter **B** and the number of asset of each of these brands:

```
SELECT Model.Brand.Name, count(lAstId) FROM amAsset GROUP BY Model.Brand.Name
HAVING Model.Brand.Name > 'B'
```

It is also possible to express the same query using the `WHERE` clause:

```
SELECT Model.Brand.Name, count(lAstId) FROM amAsset WHERE Model.Brand.Name > 'B'
GROUP BY Model.Brand.Name
```

Example of query using the `HAVING` clause:

The `HAVING` clause enables you to use aggregate functions (such as `Count`); This is not the case with the `WHERE` clause. Thus, the following query searches all brands represented by more than one asset:

```
SELECT Model.Brand.Name, count(lAstId) FROM amAsset GROUP BY Model.Brand.Name
HAVING count(Model.Brand) > 1
```

ORDER BY clause

The AQL `ORDER BY` clause is equivalent to clause with the same name in SQL.

Items can be sorted:

- In ascending order: `ASC`. This is the default sort order.
- In descending order: `DESC`.

Syntax

```
ORDER BY <expression> [ASC | DESC] [, <expression> [ASC | DESC]...]
```

INSERT clause

This clause enables you to insert one or more records into a database table.

Syntax

```
INSERT INTO <Table name> [table alias] (<Name of a field> [, <Name of a field>...] VALUES (  
<expression> [, expression]...) | AQL sub-query)
```

This clause is included in the Asset Manager API **AmDbExecAql**.

For more information about the Asset Manager API, refer to the Programmer's reference guide, chapter **Alphabetic reference**.

Example

The `INSERT` clause can simplify the code of a **Supplemental delivery information wizard**:

Wizard code not using the `INSERT` clause

```
hrAlarm = AmCreateRecord("amDateAlarm")  
    lErr = AmSetFieldLongValue(hrAlarm, "bSecondLevel", 0)  
    lErr = AmSetFieldLongValue(hrAlarm, "dtTrig1", AmGetFieldLongValue  
(hrAsset, 2)-1DaysBefore*86400)  
    lErr = AmSetFieldLongValue(hrAlarm, "lAction1Id", lActionId)  
    lErr = AmSetFieldLongValue(hrAlarm, "lMonitObjId", lAstId)  
    lErr = AmSetFieldStrValue(hrAlarm, "MonitoredField", "dWarrEnd")  
    lErr = AmSetFieldStrValue(hrAlarm, "MonitoredTable", "amAsset")
```

```
lErr = AmSetFieldLongValue(hrAlarm, "sDaysBefore1", lDaysBefore)  
lErr = AmInsertRecord(hrAlarm)
```

Wizard code using the **INSERT** clause

```
lErr = AmDbExecAql("insert into amDateAlarm (bSecondLevel, dtTrig1, lActionId,  
lMonitObjId, MonitoredField, MonitoredTable, sDaysBefore1) values ( 0, " &  
AmGetFieldLongValue(AmGetFieldLongValue(hrAsset, 2)-lDaysBefore*86400 & ", " &  
lAstId & ", 'dWarrEnd', 'amAsset', " & lDaysBefore & ")")
```

UPDATE clause

This clause enables you to update one or more fields of a record in a database table.

Syntax

```
UPDATE <table name> [table alias] SET (<name of a field> [, <name of a field>...]) [FROM clause]  
[WHERE clause]
```

Example

The **UPDATE** clause can help simplify the code of an action that triggers an order action :

Action code not using the **UPDATE** clause

```
hr = AmGetRecordFromMainId("amPOrder", [lPOrdId])  
lErr = AmSetFieldLongValue(hr, "seStatus", "$(IDS_POSTSTATUS_ORDERED)")  
lErr = AmUpdateRecord(hr)
```

Action code using the **UPDATE** clause

```
lErr = AmDbExecAql("update amPOrder set seStatus = 21 where lPOrdId = " &  
[lPOrdId])
```

DUPLICATE clause

This clause enables you to duplicate a record existing in a database table.

This function is particular to Asset Manager.

For more information, refer to the **User interface** guide, chapter **Operations on records**, section **Duplicating a record**.

Syntax

DUPLICATE <table name> [table alias] SET (<name of a field> [, <name of a field>...]) [FROM clause]
[WHERE clause]

DELETE clause

This clause enables you to delete the fields of a record in a database table.

Syntax

DELETE [FROM clause] [WHERE clause]

AQL function reference

The following AQL functions can be used in queries and formulas:

- Aggregate type AQL functions
- String type AQL functions
- Date type AQL functions
- Numeric type AQL functions
- Test type AQL functions

Note: You can also use native SQL functions of your DBMS. In this case, the resulting code is not portable.

This section includes:

Aggregate-type AQL functions	61
String-type AQL functions	61
Date-type AQL functions	63
Numeric-type AQL functions	65
Test type AQL functions	67

Aggregate-type AQL functions

AQL - Aggregate-type functions

Function	Description
Avg(<column>)	Returns the average value of a "number" type column. Returns "0" if the column does not have any records.
Count(<Column>)	Counts the non-null values in a column.
Countdistinct (<Column>)	Counts the distinct non-null values in a column.
Max(<column>)	Returns the maximum value of a "Number", "Text" or "Date" type column. If the column does not have any records, returns "0" ("number" type column), "empty string" ("Text" type column), or "empty date" ("Date" type column).
Min(<Column>)	Returns the minimum value in a "Number", "Text" or "Date" type column. If the column does not have any records, returns "0" ("Number" type column), "empty string" ("Text" type column), or "empty date" ("Date" type column).
Sum(<Column>)	Returns the sum of the values of a "number" type column. Returns "0" if the column does not have any records.
Sumdistinct(<Column>)	Returns the distinct sum of the values of a "number" type column. Returns "0" if the column does not have any records.

These functions jointly use the "GROUP BY" and "HAVING" clauses.

String-type AQL functions

AQL - String-type functions

Function	Description
Ascii(<String>)	Returns the ASCII value of the first character of the <string>.
Char(<n>)	Returns the character with ASCII code "n".
Length(<String>)	Returns the length of the <string>.
CharIndex(<String1> ,	Returns the position of <string1> inside <string2>. The first character in <string1> is at position 1. If there is no occurrence of <string1> in <string2>

AQL - String-type functions, continued

Function	Description
<String2>)	then the function returns 0.
DateToText(<Date>)	Converts a <date> into a text data type (a string that describes an ISO international formatted date)
EmptyString()	Creates a non-null empty string.
LikeParam(<Field>)	Used when comparing two fields with the AQL operand Like. Example: Field1 like LikeParam(Field2)
Left(<String>, <n>)	Returns the "n" first characters of the <string>.
Lower(<String>)	Returns the <string> in lowercase.
Ltrim(<String>)	Removes the spaces at the left of the <string>.
NullBlob()	Creates a NULL Blob. Note: Programmatically it represents a NULL casted into a Blob object.
NullMemo()	Creates a NULL Memo. Note: Programmatically it represents a NULL casted into a Memo object.
NullString()	Creates a NULL String. Note: Programmatically it represents a NULL casted into a String object.
NumberToText(<Number>)	Converts a <number> into a text data type.
Right(<String>, <n>)	Returns the "n" last characters of the <string>
Rtrim(<String>)	Removes the spaces at the right of the <string>
Substring(<String>, <n1>, <n2>)	Extracts the sub-string starting at character "n1" in the <string> and with length "n2" (the 1st character of the <string> is considered as character number 1).
TimeStampToText (<Date+Time>)	Converts a Date+Time type object into a text data type that conforms to the following international standard: YYYY-MM-DD HH24:MI:SS
TimeToText(<Time>)	Converts a Time type object into a text data type that conforms to the following international standards: HH:MM:SS

AQL - String-type functions, continued

Function	Description
Upper(<String>)	Returns the <string> in uppercase.

Date-type AQL functions

AQL - Date-type functions

Function	Description
AddDays(<date>, <number>)	Adds a given number of days to a "Date" or "Date and time" type field.
AddHours(<date>, <number>)	Adds a given number of hours to a "Date" or "Date and time" type field.
AddMinutes(<date>, <number>)	Adds a given number of minutes to a "Date" or "Date and time" type field.
AddSeconds(<date>, <number>)	Adds a given number of seconds to a "Date" or "Date and time" type field.
Day(<date>)	Returns the number of the day in the month for a "Date" or "Date and time" type field (1-31).
DayOfYear(<date>)	Returns the number of the day in the year for a "Date" or "Date and time" type field (1-366).
DaysDiff(<date1>, <date2>)	Number of days between the dates date1 and date2 ("floating point" number with decimals)
DbToLocalDate (<date>)	Converts a date expressed in the time zone of the database server to a date expressed in the time zone defined at client machine level.
Getdate()	Returns the server's current system date.
Hour(<time>)	Returns the number of the hour in the day for a "Time" or "Date and time" type field (0-23).
HoursDiff(<date1>, <date2>)	Number of hours between the dates date1 and date2 ("floating point" number with decimals) Note: The HoursDiff function returns different results depending on which the

AQL - Date-type functions, continued

Function	Description
	<p>database used, due to how the database rounds minutes. Example: 59 minutes in DB2 = 0 hours, whereas 59 minutes in MSSQL = 1 hour.</p>
LocalToDbDate(<date>)	Converts a date expressed in the time zone of the client machine to a date expressed in the time zone of the database server.
Minute(<time>)	Returns the number of minutes for a "Time" or "Date and time" type field (0-59).
MinutesDiff(<date1>, <date2>)	Number of minutes between the dates date1 and date2 ("floating point" number with decimals)
Month(<date>)	Returns the number of the month for a "Date" or "Date and time" type field (1=January, ..., 12=December).
NullDate()	Creates a NULL Date. Note: Programmatically it represents a NULL casted into a Date object.
NullTime()	Creates a NULL Time. Note: Programmatically it represents a NULL casted into a Time object.
NullTimeStamp()	Creates a NULL Date and time. Note: Programmatically it represents a NULL casted into a Date and Time object.
NumberToTime(<number>)	Converts a number and to a "Date and time"-type date.
Second(<time>)	Returns the number of seconds for a "Time" or "Date and time" type field (0-59).
SecondsDiff(<date1>, <date2>)	Number of seconds between the dates date1 and date2 ("floating point" number with decimals)
TextToTime(<text>, <format>, <language>)	Converts text to a date. The additional (and optional) language and format parameters give access to database-specific formats and make it possible to specify the expected format for conversion and the language support settings. For further information, refer to your database documentation.
WeekDay(<date>)	Returns the number of the day in the week for a "Date" or "Date and time" type field. This number depends on how the server is configured. For example, the default configuration under Microsoft SQL Server is (1=Sunday, 2=Monday, ...,

AQL - Date-type functions, continued

Function	Description
	7=Saturday). The default configuration under Oracle is (1=Monday, ..., 7=Sunday).
Year(<date>)	Returns the number representing the year for a "Date" or "Date and time" type field (e.g: 2000).

Examples of Date-type functions

Description	Asset Manager query language
All records modified during the last week	AddDays(dtLastModif,7)>=Getdate()
All work orders notified in the last hour	HoursDiff(Getdate(), dtNotif) <= 1 or AddHours(dtNotif, 1) >= Getdate()
All work orders closed in the last half-hour	MinutesDiff(Getdate(), dtActualFixed) <= 30 or AddMinutes(dtActualFixed, 30) >= Getdate()

The following query lists the work orders effectively carried out and resolved during the same day. The time zone of the client machine is taken into account:

```
SELECT Self FROM amWorkorder WHERE DayOfYear(DbToLocalDate(dtActualFixStart)) =
DayOfYear(DbToLocalDate(dtActualFixed))
```

The following query lists all work orders that have effectively been started today:

```
SELECT Self FROM amWorkorder WHERE DayOfYear(DbToLocalDate(dtActualFixStart)) =
DayOfYear(DbToLocalDate(GetDate()))
```

Numeric-type AQL functions

AQL - Numeric-type functions

Function	Description
Abs(<Number>)	Returns the absolute value of a "number".
Ceil(<Number>)	Returns the smallest integer greater or equal to a "number".
DataLength(<Data>)	Returns the <Data> length in bytes.
Floor(<Number>)	Returns largest integer less than or equal to a "number".

AQL - Numeric-type functions, continued

Function	Description
Length (<Data>)	Returns the <Data> length in characters.
Mod(<a>,)	Returns the remainder of the division of "a" by "b" ($a = qb + r$, with q integer and $0 < r < q$).
NullNumeric()	Creates a NULL Numeric. Note: Programmatically it represents a NULL casted into a Numeric object.
NumberToNumber(<Number>)	Allows the conversion between numbers of different types if it is not performed at the RDBMS level.
Round(<a>, <n>)	Rounds "a" to "n" decimal places.
Sign(<Number>)	Enables you to determine the sign of the <Number> parameter: <ul style="list-style-type: none"> • If the function returns 1, the <Number> is positive. • If the function returns -1, the <Number> is negative. • If the function returns 0, the <Number> is null (=0).
TextToNumber (<Text>)	Converts a <Text> into a numeric data type.
Trunc(<a>, <n>)	Truncates "a" to "n" decimals.

Examples of application:

Abs (2.516) = 2.

Ceil (2.516) = 3.

Floor (2.516) = 2.

Mod (6,4) = 2.

Round (31.16, 1) = 31.20.

Round (31.16, 0) = 31.00.

Round (31.16, -1) = 30.00.

Trunc (31.16, 1) = 31.1.

Test type AQL functions

AQL - Test-type functions

Function	Description
IsNull(<a>,)	If "a" is "Null", replaces "a" by "b". The types of "a" and "b" must be compatible.
AQLdcode (<expression>, <value>, <trueresult>, <falseresult>)	<p>If <expression> equals to <value>, <trueresult> is returned. Otherwise, <falseresult> is returned. The value of <expression> and <value > must be the same data type.</p> <p>For example, the statement <code>AQLDecode(bCompact, 0, (PortfolioItem.fQty * lUseCount), lUseCount)</code> is similar to:</p> <pre>If bCompact =0 then result =(PortfolioItem.fQty * lUseCount) else result = lUseCount endif</pre>

Examples of queries

Each of the following examples deal with a specific aspect of query design. You can modify or combine these examples to use them as the basis of your own queries.

These examples give the full syntax of the query. If you want to test them out as is, use Asset Manager Export Tool. You will have to modify the syntax of these examples to use them in a query filter, for example.

Thus, the following query given in full:

```
SELECT self FROM amAsset WHERE Model.Brand.Name='Compaq'
```

is written as follows when used in a query filter (only the WHERE clause is explicitly given) used on the table of assets:

```
Model.Brand.Name='Compaq'
```

Further examples of queries are included in the demonstration database supplied with Asset Manager.

Note: To view the transcription of a query in the corresponding SQL code of the DBMS you are

using, display the **Preview** tab in the query detail.

This section includes the following examples:

To compare a field in the main table with a value	68
To compare a link in the main table with another link	68
To compare a link in the main table with a value	68
To compare according to a field in a table linked to the main table	69
Hierarchic tables	69
Query combining two conditions	69
Comparison of a field with numbers, dates or text	70
Query concerning a feature	70
To search records according to an expression	70
To search a field that is not populated	70
To search for the absence of a link	70
Query with alias	72
Query with variable	72

To compare a field in the main table with a value

Example: All "Compaq" brand portfolio items.

```
SELECT Self FROM amPortfolio WHERE Model.Brand.Name = 'Compaq'
```

To compare a link in the main table with another link

Example: All portfolio items with the same location as their parent asset.

```
SELECT Self FROM amPortfolio WHERE Location = Parent.Location
```

To compare a link in the main table with a value

Example: All departments and employees directly linked to the "Burbank Agency".

```
SELECT Self FROM amEmplDept WHERE Parent.Name = 'Burbank Agency'
```

To compare according to a field in a table linked to the main table

Example: All portfolio items that have the same location name as their parent.

```
SELECT Self FROM amPortfolio WHERE Location.Name = Parent.Location.Name
```

Hierarchic tables

Utilization of "FullName" field

Example: All sub-locations of the location named "Ariane Building":

```
SELECT Self FROM amLocation WHERE FullName LIKE '/Ariane Building/%'
```

Utilization of "FullName" and "sLvl" fields

Queries on the hierarchic tables often make use of the "FullName" and "sLvl" fields.

Example: All the sub-locations of the location "Ariane Building", with a hierarchic level less than 3.

The hierarchic value of the root level of a tree-structure is equal to "0".

```
SELECT Self FROM amLocation WHERE (FullName LIKE '/Ariane Building/%') AND (sLvl < 3)
```

Pay special attention to the "/" characters that appear at the start and end of full names.

Query combining two conditions

Example: All employees with title "Account executive" and located at the "Burbank site".

```
SELECT Self FROM amEmplDept WHERE (Title = 'Commercial') AND (Location.Name = 'Madison site')
```

Comparison of a field with numbers, dates or text

For example: All work orders carried out between January 1, 2006 and December 31, 2006.

```
SELECT self FROM amWorkOrder WHERE (dtActualFixStart >= #2006-01-01 00:00:00#) AND  
(dtActualFixStart <= #2006-12-31 00:00:00#)
```

Query concerning a feature

Example: All portfolio items whose feature with SQL name **fv_Size** showing a value greater than or equal to 150 cm.

```
SELECT Self FROM amPortfolio WHERE fv_Size >= 150.00
```

To search records according to an expression

Example: All assets whose purchase price is equal to the greatest purchase price contained in the database. Note that a sub-query is used in the main query to identify the maximum price.

```
SELECT Self FROM amAsset WHERE mPrice = (SELECT max(mPrice) FROM amAsset)
```

To search a field that is not populated

Example: All employees without a telephone number. Note that an empty string is represented by two single quotes ' '.

```
SELECT Self FROM amEmpDept WHERE Phone=' '
```

To search for the absence of a link

Case of a 1 link

Example: All portfolio items that have not been assigned to a user. Note that the absence of a link is

denoted by "0".

```
SELECT Self FROM amPortfolio WHERE User = 0
```

Case of n links

Example: All models with no associated assets:

```
SELECT self FROM amModel WHERE NOT ( EXISTS (SELECT A1.IAstId FROM amAsset A1 WHERE  
A1.IModelId = amModel.IModelId))
```

This query scans the Models table, takes each model one after the other and compares the number of assets belonging to this model with 0.

Example combining a test on a 1 link and an n link

Example: All models without parent model or sub-model:

```
SELECT self FROM amModel WHERE (NOT ( EXISTS (SELECT A1.IModelId FROM amModel A1  
WHERE A1.IParentId = amModel.IModelId))) AND (Parent = 0)
```

This query performs:

- A test on a 1 link ("Parent = 0"), to select those models without parent assets.
- A test in an n link ("0 = (SELECT COUNT(a.IModelId) FROM amModel a WHERE a.IParentId = IModelId)", to select those models without sub-models. The test on the n link takes each model, selects its identifier "IModelId", and counts all the models whose "IParentId" identifier is equal to "IModelId".

Note: The SELECT COUNT clause counts all database records; It is therefore more costly in terms of performance than the EXIST clause.

Another example

All models without "Computer" natured sub-model:

```
SELECT self FROM amModel p WHERE NOT ( EXISTS (SELECT IModelId FROM amModel WHERE  
(FullName LIKE (p.FullName + '%/')) AND (Nature.Name = 'Computer')))
```

Note: If you try this query with Asset Manager Export Tool, an error message will appear. You can ignore this message. The query actually works correctly.

Query with alias

Example: All employees having taken the 'HPE Software' training program and the 'Database' training program.

Starting table: The table of departments and employees.

The query is as follows:

```
SELECT Self FROM amEmplDept WHERE (Trainings_1.Title = 'HPE Software') AND  
(Trainings_2.Title = 'Database')
```

Aliases, expressed as "Training_1" and "Training_2" enable you to define 2 conditions concerning the 2 different records linked by the "Training" link.

If we had written:

```
SELECT Self FROM amEmplDept WHERE (Trainings.Title = 'HPE Software') AND  
(Trainings.Title = 'Database')
```

We would have selected all employees having taken a training course with both titles.

If we had written:

```
SELECT Self FROM amEmplDept WHERE (Trainings.Title = 'HPE Software') OR  
(Trainings.Title = 'Database')
```

We would have selected all employees having taken one training course with one of the two titles.

You can use the colon character to shorten your query:

```
SELECT amPortfolio.self, amModel_FullName:self FROM amPortfolio portfolio
```

This query is equivalent to:

```
SELECT amPortfolio.self, FullName.semf FROM amPortfolio portfolio, amModel FullName
```

Query with variable

Syntax 1

For example: All models whose name is equal to a value that is specified by the user when the user selects a filter that uses the query.

Starting table: **Models** (amModel).

Name = \$Nom

The query filter user is requested to populate the **Name** field before executing the query.

If the user enters **Test**, the query that is generated will be:

```
SELECT M1.Name FROM amModel M1 WHERE M1.Name = 'Test'
```

Syntax 2

For example: All models whose name contains a value that is specified by the user when the user selects a filter that uses the query.

Starting table: **Models** (amModel).

Name LIKE amDbVal(Name, 'Name contains', 'Specify the value that the field must contain')

The query filter user is requested to populate the **Name contains** field before executing the query.

The **Name contains** field displays **Specify the value that the field must contain** by default.

If the user enters **Test**, the query that is generated will be:

```
SELECT M1.Name FROM amModel M1 WHERE M1.Name LIKE 'Test' ESCAPE '\'
```

Chapter 4: SAP Crystal Reports

This chapter explains how to display and print reports with Asset Manager using SAP Crystal Reports.

Note: The SAP Crystal Reports are not available for the Linux version of Asset Manager.

This chapter includes:

General overview	74
Installing and configuring to enable Asset Manager clients to access reports stored in the SAP BusinessObjects Business Intelligence Platform 4.1	76
Migrating from BusinessObjects Enterprise XI 3.1 to BusinessObjects Business Intelligence Platform 4.1	100
Display a report	100
Making reports available	104
How to modify a Seagate Crystal report	108
Creating a detail report	109
Printing a report	111
Identifying Crystal Reports specific to a given module	113
Associating a report with a button in a screen (Windows client) or with actions in a screen (Web client)	114
Limitations	114

General overview

This section includes:

Why use SAP Crystal Reports (Designer)?	75
Where are the reports stored?	75
Where are the reports referenced?	75
Report types	75

Why use SAP Crystal Reports (Designer)?

SAP Crystal Reports (Designer) is used to create and modify reports (.rpt files).

You can view and print the reports created with SAP Crystal Reports (Designer) using Asset Manager Windows and Web client; however, you cannot create or modify the reports with Asset Manager.

Where are the reports stored?

.rpt files can be stored in different ways:

- In the **Asset Manager database**.
Reports stored in this manner can only be accessed via the Windows client.
- In the **SAP BusinessObjects Business Intelligence Platform**.
Reports stored in this manner can be accessed via the Windows and the Web clients.
When a user requests a SAP Crystal Reports report (display or print-out) via a Asset Manager Web or Windows client, a query is sent to SAP BusinessObjects Business Intelligence Platform.
SAP BusinessObjects Business Intelligence Platform retrieves the data from the Asset Manager database and displays the report in the Windows client or an Internet Explorer (Web client).

Where are the reports referenced?

Each .rpt file is referenced by a record in the **Reports** (amReport) table, regardless of how it is stored:

- If the report is stored in the **Asset Manager database**: The .rpt file is imported into the database via the Windows client (**Import** button located on the reports detail page).
- If the report is stored in the **SAP BusinessObjects Business Intelligence Platform**: The .rpt file is referenced via the **File** (FileName) field located on the reports detail page.

Report types

There are several report types:

- **Detail reports**

These reports are used to display information about a record from a given table.
These reports can only be accessed if a record from the report's context table is selected.
These reports are only available for Windows clients.

- **List reports and graphs**

These reports are used to display information on a selection of records from a given table.
These reports can be accessed at all times.
These reports are available for Windows and Web clients.

Installing and configuring to enable Asset Manager clients to access reports stored in the SAP BusinessObjects Business Intelligence Platform 4.1

This section includes:

Prerequisites	76
Installing SAP BusinessObjects Business Intelligence Platform 4.1 and SAP Crystal Reports (Designer)	77
Configuration checklist for unattended and manual installation	81
Configurations to enable Asset Manager Web clients to access reports stored in the BusinessObjects Business Intelligence Platform database	86
Configuring SSO	93
Declaring the access URL to the SAP BusinessObjects Business Intelligence Platform	97
Mapping the report file name with Crystal report ID	98
Configuring the sysCoreWebCrystal calculated field	99

Prerequisites

Create a connection to the Asset Manager database

To enable the SAP BusinessObjects Business Intelligence Platform to access Asset Manager database, you need to create an ODBC connection with an Asset Manager ODBC driver.

If Asset Manager and BusinessObjects Business Intelligence Platform are not deployed on the same computer, you must install the following two components on the computer that is running BusinessObjects Business Intelligence Platform.

- Asset Manager Windows client
- Asset Manager ODBC driver

See Asset Manager **Installation and upgrade** guide and Asset Manager **Administration** guide, chapter **Accessing the database via ODBC**.

In addition, follow these steps to create the ODBC connection.

1. Start the Asset Manager Windows client.
2. Create a connection to the Asset Manager database (**File/ Manage connections...** menu).
See Asset Manager **Administration** guide, chapter **Creating, modifying and deleting an Asset Manager database**, section **Creating a connection with Asset Manager**.
Remember the name of the connection. You will need it later.
3. Test the connection (**Test** button in the Manage connections window).

Installing SAP BusinessObjects Business Intelligence Platform 4.1 and SAP Crystal Reports (Designer)

SAP BusinessObjects Business Intelligence Platform and SAP Crystal Reports (Designer) can be installed in several ways:

- Via the **OEM version** (Original Equipment Manufacturer) which is provided in the **AssetManager_CRS_9.50.zip** and **AssetManager_CRD_9.50.zip** files.

You can perform an unattended or manual installation.

- Via a **full commercial version** that you already own.

See SAP Crystal Reports documentation.

To find out which SAP Crystal Reports versions are supported, see Asset Manager Support Matrix.

Note: Settings will be set for each of the reports.

Note: Only SAP Crystal Reports installed on Windows is supported for use with Asset Manager.

This is because SAP Crystal Reports uses the Asset Manager ODBC driver.

This section describes the installation of the OEM version of SAP Crystal Reports (Designer) for Asset Manager and SAP BusinessObjects Business Intelligence Platform for Asset Manager from the **AssetManager_CRD_9.50.zip** and **AssetManager_CRS_9.50.zip** files that are provided.

Installing SAP Crystal Reports (Designer) and SAP BusinessObjects Business Intelligence Platform for Asset Manager (unattended mode)

The unattended mode installation is carried out automatically using predefined options.

To install the SAP BusinessObjects Business Intelligence Platform, follow these steps.

1. Launch the **AssetManagerReportingInstall.bat** script located at the root of the **AssetManager_CRS_9.50.zip** file that is provided.

Note: You must use the "Run as administrator" option to run this script.

2. You will be provided with the following language version options.

Please choose one of the options below:

- 1- Install English version
- 2- Install German version
- 3- Install Spanish version
- 4- Install French version
- 5- Install Russian version
- 6- Install Italian version
- 7- Install Japanese version
- 8- Install Korean version
- 9- Install Simplified Chinese version
- 10- Install Netherlands version
- 11- Install Portuguese version
- 0- Exit

Your choice:

Enter the digit which represents the version you want to install.

3. You have two choices:

Asset Manager Reporting Installer

=====

Please choose one of the options below:

1- Install SAP BusinessObjects Business Intelligence Platform for Asset Manager

0- Exit

Your choice:1

Enter '1' to proceed with the installation.

4. Choose the installation mode when you are provided with the following options.

```
Asset Manager Reporting Installer
```

```
=====
```

```
Install SAP BusinessObjects Business Intelligence Platform for Asset Manager:
```

```
  1- Unattended installation
```

```
  2- Manual installation
```

```
  0- Back
```

Your choice:1

Enter '1' to select unattended installation mode.

5. Type to create the administrator password of the BusinessObjects Business Intelligence Platform.
6. The installation will be carried out using the predefined options. The installation process will halt at different times and will need to be continued manually.
7. Enter the administrator password you set in step 5 to import the Crystal Reports for Asset Manager.

If you want to create or modify the Crystal Reports, you can install SAP Crystal Reports (Designer) for Asset Manager by running the `AssetManagerReportingInstall.bat` file at the root of the **AssetManager_CRD_9.50.zip**. The installation steps of SAP Crystal Reports (Designer) are similar to the steps above.

Installing SAP Crystal Reports (Designer) and SAP BusinessObjects Business Intelligence Platform for Asset Manager manually

If you need to customize your installation of SAP BusinessObjects Business Intelligence Platform for Asset Manager and/or SAP Crystal Reports (Designer) for Asset Manager, you will need to perform a manual installation. For example, you can use the manual installation to customize the CMS server connection port, Tomcat ports, database type, and so on.

For more information, read the BusinessObjects guides available at:

http://help.sap.com/businessobject/product_guides/.

If you cannot access the documentation server, you will find the following documents on SAP BusinessObjects Business Intelligence Platform in the **<Asset Manager installation media>:\Doc** folder.

- Release Notes
- Installation Guide (English version)

To perform a manual installation of SAP BusinessObjects Business Intelligence Platform for Asset Manager or SAP Crystal Reports for Asset Manager:

1. Launch the **AssetManagerReportingInstall.bat** script located at the root of the **AssetManager_CRS_9.50.zip** file that is provided.
2. You will be provided with the following language version options.

Please choose one of the options below:

- 1- Install English version
- 2- Install German version
- 3- Install Spanish version
- 4- Install French version
- 5- Install Russian version
- 6- Install Italian version
- 7- Install Japanese version
- 8- Install Korean version
- 9- Install Simplified Chinese version
- 10- Install Netherlands version
- 11- Install Portuguese version
- 0- Exit

Your choice:

Enter the digit which represents the version you want to install.

3. You have two choices:

```
Asset Manager Reporting Installer
```

```
=====
```

Please choose one of the options below:

1- Install SAP BusinessObjects Business Intelligence Platform for Asset Manager

0- Exit

Your choice:1

Enter '1' to proceed with the installation.

4. Choose the installation mode when you are provided with the following options.

```
Asset Manager Reporting Installer
```

```
=====
```


Install SAP BusinessObjects Business Intelligence Platform for Asset Manager:

- 1- Unattended installation
- 2- Manual installation

0- Back

Your choice:2

Enter '2' to select Manual installation mode.

5. Next, follow the procedure that is displayed by the installation wizard.

For more information about the SAP installation procedure, see Business Intelligence platform Installation Guide for Windows on <http://help.sap.com/bobip>.

If you want to create or modify the Crystal Reports, you can install SAP Crystal Reports (Designer) for Asset Manager by running the `AssetManagerReportingInstall.bat` file at the root of the **AssetManager_CRD_9.50.zip**. The installation steps of SAP Crystal Reports (Designer) are similar to the steps above.

Configuration checklist for unattended and manual installation

The **manual** installation of SAP BusinessObjects Business Intelligence Platform for Asset Manager requires that additional configuration procedures be performed manually. The following table lists the checklist of the configurations for your reference.

If some configurations were not automatically implemented in the unattended installation mode, you can check the following this table and implement them manually.

Note: No matter what mode you choose, read the full chapter before you start.

	Where to check the configuration	Criteria	Whether it is automatically implemented in unattended installation	Manual implementation process steps
1. The <code>u2lamlib.dll</code>	<Full path to	The <code>u2lamlib.dll</code> file	✓	See " Importing the

	Where to check the configuration	Criteria	Whether it is automatically implemented in unattended installation	Manual implementation process steps
file	the BusinessObjects installation folder>\SAP BusinessObjects Enterprise <version>\win32_x86	is imported to this folder		u2lamlib.dll file (required for both unattended and manual installations) " on page 86.
2. The demo reports shipped with Asset Manager 9.61 is imported to the BusinessObjects Business Intelligence Platform	Folders screen of the SAP BusinessObjects Central Management Console	An Asset Manager folder including the demo reports is created.	✓ ¹	Reports can be imported with LCMBIAR file or individual .rpt files. See "Importing LCMBIAR file by SAP Promotion Management Tool" on page 87. See "Importing Asset Manager .rpt files manually" on page 88.
3. The user account for Asset Manager has been created in the BusinessObjects Business Intelligence Platform	Users and Groups screen of the Central Management Console	An AssetManager user with Full Control access level is created.	✓ ²	See "Creating the Asset Manager integration user in SAP BusinessObjects Business Intelligence Platform (required for manual installation)" on page 88.
4. The data source of the Asset Manager reports is specified	Database Configuration page of the Asset Manager folder default settings	The Use custom database logon information specified here	✓ Note: You need to	See "Set the report property - "Database Configuration"

	Where to check the configuration	Criteria	Whether it is automatically implemented in unattended installation	Manual implementation process steps
		<p>option is selected, and the data source properties are as follows:</p> <ul style="list-style-type: none"> • Database Type: Select a database driver, ODBC • Server: Asset Manager Databases • Database: For example, AMDemo95en • User: Admin • Table Prefix: Use default table prefix • When viewing report: Prompt the user for database logon (this option is required by the single sign-on mechanism) 	<p>manually update the Database property.</p>	<p>(required for both unattended and manual installations)" on page 90.</p>
<p>5. The single sign-on has been implemented</p>	<p><Installation folder of BusinessObjects Business Intelligence Platform Web application server>\lib</p>	<p>The am-reporting.common-96.jar and am-reporting.serverside-96.jar files are imported into this folder.</p>	<p>✓ Note: You need to manually update the web.xml of Asset Manager</p>	<p>See "Configuring SSO" on page 93.</p>

	Where to check the configuration	Criteria	Whether it is automatically implemented in unattended installation	Manual implementation process steps
	<p>web.xml of Asset Manager Web tier</p> <p>The SSO filter section is updated.</p>		<p>Web tier and "Configure the password for single sign-on:" on page 96.</p>	
	<p>web.xml of BusinessObjects Business Intelligence Platform</p> <p>The SSO filter section is added.</p>			
	<p><Installation folder of BusinessObjects Business Intelligence Platform Web application server>\WEB-INF\classes</p> <p>The reporting.properties file is imported into this folder.</p>			
6. The sysCoreWebCrystal calculated field has been configured	<p>The Calculated fields screen of Asset Manager client (Administration/system/Calculated fields navigation link)</p>	<p>The Display report (sysCoreWebCrystal) calculated field is updated.</p>		<p>See "Configuring the sysCoreWebCrystal calculated field" on page 99.</p>
7. The URL address of the application server hosting the SAP BusinessObjects Business Intelligence Platform has been declared	<p>Edit the URL address of the application servers... wizard (BstBackEndOpt) (Administration/User actions/Edit the URL address of the</p>	<p>The Server's URL address of SAP BusinessObjects Crystal Reports Server is http or https://<SAP BusinessObjects Business Intelligence Platform server</p>		<p>See "Declaring the access URL to the SAP BusinessObjects Business Intelligence Platform" on page 97.</p>

	Where to check the configuration	Criteria	Whether it is automatically implemented in unattended installation	Manual implementation process steps
	application servers... (navigation link)	name>:<Port used by SAP BusinessObjects Business Intelligence Platform>.		
8. The report File name in Asset Manager has been mapped with the report ID in the SAP BusinessObjects Business Intelligence Platform	Reports screen (Administration/ Reporting/ Reports navigation link)	The File field is populated with the value of the report ID in the Properties page of the report's default settings.		See " Mapping the report file name with Crystal report ID " on page 98.
<ul style="list-style-type: none"> • ¹: In the unattended installation mode, the Asset Manager reports are automatically imported by deploying a LCMBIAR (Business Intelligence Archive Resource) file in the SAP BusinessObjects Business Intelligence Platform database using the tool_step2_deployReports.bat script (located in the install folder). Therefore, in the rare circumstance that the reports are not successfully reported, you can manually run the tool_step2_deployReports.bat to re-import the reports. • ²: In the unattended installation mode, the AssetManager user is automatically created; however, you need to manually grant full-control rights to the user by referring to "Assigning rights" on page 89. 				

Configurations to enable Asset Manager Web clients to access reports stored in the BusinessObjects Business Intelligence Platform database

If you performed an unattended mode installation of SAP BusinessObjects Business Intelligence Platform for Asset Manager from the provided OEM version, you do not need to follow every configuration procedure described in this section (you only need to manually set the Asset Manager web client and domain configurations).

Find the procedures required for unattended installation in "[Configuration checklist for unattended and manual installation](#)" on page 81.

However, if you performed a manual installation of SAP BusinessObjects Business Intelligence Platform for Asset Manager from the provided OEM version or if you installed a full commercial version of SAP BusinessObjects Business Intelligence Platform, the following configuration steps are required by the integration.

Importing the **u2lamlib.dll** file (required for both unattended and manual installations)

- If Asset Manager and SAP BusinessObjects Business Intelligence Platform have been installed on the same server, no configuration is required for the **u2lamlib.dll** file.
- If Asset Manager and SAP BusinessObjects Business Intelligence Platform are not installed on the same server:
 - a. Unzip the **AssetManager-CRS-9.50.zip** file and open it.
 - b. Display the contents of the **lib\EN** folder.
 - c. Copy the **u2lamlib.dll** file.
 - d. Paste this file in the **<Full path to the BusinessObjects installation folder>\SAP BusinessObjects Enterprise <version>\win32_x86** folder.
For example:

```
C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI
4.0\win32_x86
```

Importing Asset Manager reports into SAP BusinessObjects Business Intelligence Platform (required for manual installation)

Asset Manager reports can be imported into SAP BusinessObjects Business Intelligence Platform in different ways.

Importing reports with the old BIAR file to BusinessObjects Business Intelligence Platform 4.1

BIAR archive files that were created by the previous SAP BI products can no longer work in BusinessObjects Business Intelligence Platform 4.x. Instead, BusinessObjects Business Intelligence Platform uses LCMBIAR archive files.

Therefore, if you have existing BIAR files of the Asset Manager reports and you want to upgrade to BusinessObjects Business Intelligence Platform 4.1, you need to run SAP Upgrade Management Tool to import reports that are contained in BIAR files. For more information about SAP Upgrade Management Tool, see <http://scn.sap.com/community/bi-platform/blog/2013/04/16/bi4-upgrade-management-tool>.

After you import the BIAR file, you can find the Asset Manager reports in the **Asset Manager** folder.

Importing LCMBIAR file by SAP Promotion Management Tool

In an Asset Manager version that is compatible with BusinessObjects Business Intelligence Platform 4.x (for example, Asset Manager 9.50 Patch 4), a LCMBIAR file named `am_reports_<language>.lcmbar` is provided in the Crystal Report installation folder. This file contains all the out-of-box Asset Manager reports and relevant information. To import this LCMBIAR file, follow these steps.

1. Log on to Central Management Console, open the Promotion Management Tool.
2. Click **Import**, and then click **Import file**.
3. Click Browse, select the `am_reports_<language>.lcmbar` file in the Crystal Report installation folder, for example, `C:\AssetManager_CRS_9.50\lib\reports`, and then click **OK**.
4. Specify the value for the **Destination** field, and then click **Create**. A job is created.
5. On the new job page, click **Promote**.

6. On the **Security settings** tab, select **Promote Security**, enable all three check boxes, click **Save**.
7. On the **Test Promote** tab, click **Test Promote**. Make sure that the test result meets your expectation.
8. On the **Summary** tab, click **Promote**.
9. Refresh the page and observe the status of the job.
10. After the status becomes **Success**, click the **Folders** tab.
11. Click **All folders**, open **Asset Manager**, on the **User Security**, make sure the **Asset Manager** user has full control access.
12. You can find all the reports in the **Asset Manager** folder.

Importing Asset Manager .rpt files manually

You can also import individual Asset Manager reports (.rpt files) into SAP BusinessObjects Business Intelligence Platform. To do this, SAP Crystal Reports Designer must be installed and have access to the server hosting Asset Manager (to access the reports).

In this case:

1. Start the SAP Crystal Reports Designer **Report Update Wizard (Start/ All Programs/ SAP Business Intelligence/ SAP Crystal Reports 2013/ Report Upload Wizard** Windows menu).
2. Complete the wizard by selecting the reports to import (<**Asset Manager installation folder>\Datakit\bestprac\reports\rpt**) and by specifying the location (create an **Asset Manager** folder to put the reports) in the SAP BusinessObjects Business Intelligence Platform where you want to put them.

Creating the Asset Manager integration user in SAP BusinessObjects Business Intelligence Platform (required for manual installation)

- If you imported reports using the provided LCMBIAR archive file, then the report security user named "Asset Manager" has been imported to SAP BusinessObjects Business Intelligence Platform.
- Otherwise, you can create the integration user manually, see ["Manual creation" below](#).

Manual creation

Manual creation of the Asset Manager SAP BusinessObjects Business Intelligence Platform integration user has two steps:

1. **Creating the user**

To create the Asset Manager SAP BusinessObjects Business Intelligence Platform integration user:

- a. Launch the Central Management Console.
- b. Select the **Enterprise** authentication type and log on with the **Administrator** account (the password was defined during the installation process).
- c. Click the **Users and Groups** link in the **Organize** group.
- d. Select **Manager/ New/ New User** in the **Users and Groups** screen.
- e. Populate the properties as following:
 - Account Name: AssetManager
 - Full Name: AssetManager
 - Password: enter and confirm the password for AssetManager account
 - Select **Password never expires**.
 - Deselect **User must change password at next logon**.
 - Deselect **User cannot change password**.
 - Connection Type: Concurrent User
- f. Click **Create**. You will be prompted that the AssetManager account is created.

2. Assigning rights

Asset Manager reports must have already been imported into SAP BusinessObjects Business Intelligence Platform.

To assign all rights for Asset Manager reports to the AssetManager user:

- a. Launch the Central Management Console.
- b. Click **Folders** (in the **Organize** group).
- c. Right-click the folder in which you have imported your Asset Manager reports (the folder name is specified when you importing the reports).
- d. Select **User Security** from the shortcut menu.
- e. Click **Add Principals**.
- f. Select **AssetManager AssetManager** from the **Available users/groups** box.

- g. Click > to add it to the **Selected users/groups** box.
- h. Click **Add and Assign Security**.
- i. Keep the default selection for **Inheritance** and select the **Access Levels** tab in the displayed screen.
- j. Select **Full Control** in the **Available Access Levels** table.
- k. Click > to add it to the **Assigned Access Levels** table.
- l. Click **OK**.

The **AssetManager** user now has complete control of the Asset Manager reports.

Set the report property - "Database Configuration" (required for both unattended and manual installations)

The Asset Manager reports source database corresponds to the database where the report data is stored.

By default, the database value defined in the Database Configuration is "AMDemo95<language>". If you do not use the Demo database, update the Database Configuration with one of the following methods.

Modify the source database via the SAP BusinessObjects Business Intelligence Platform Central Management Console module

1. Launch the Central Management Console.
2. Click the **Folders** link in the **Organize** group.
3. Double-click the folder where you put the imported files, for example, **Asset Manager** folder.
4. Double-click the report, for example, the English demonstration database, **Analysis of levels of consumable stocks**.
5. Select **Default Settings/ Database Configuration** from the navigation menu.
6. Select **Use custom database logon information specified here**.
7. Populate the properties as following:
 - o Database Type: Select a database driver, ODBC
 - o Server: Asset Manager Databases
 - o Database: For example, AMDemo95en
 - o User: Admin

- Password: the password of your database user

Note: The default password is hardcoded **<empty>** if you have left this field blank.

- Table Prefix: Use default table prefix
- When viewing report: Prompt the user for database logon (this option is required by the single sign-on mechanism)

8. Click **Save**.

Modifying the source database via SAP Crystal Reports (Designer)

1. Start SAP Crystal Reports (Designer).
2. Open the **.rpt** file to modify.
3. On the **Database** menu, select **Set Datasource Location**.
4. Unfold the **Reports** node and select **Asset Manager Databases** from the Current Data Source box.
5. Select the the name of your database, for example, AMDemo95en. Click **Next**.
6. Populate the **User ID** field in this format: **<User ID>@<Base>**, for example: **Admin@AMDemo95en**.
7. The Asset Manager **Connect to database** dialog box appears.
Enter your password (if the user is **Admin**, the password is empty by default).
Next, click **Open**.
8. If there are any source databases for the sub-reports, perform the same operation for each of those.
9. Close the window.

Modify the source database via command line

Asset Manager reports must have already been imported into SAP BusinessObjects Business Intelligence Platform.

Note: Because the files in the zip file are read-only, you need to copy the content of the zip file to your local disk then proceed with the following steps.

To modify automatically the source database:

1. Open a DOS command and go to the **install** directory (or the corresponding local directory if you have copied the content of the zip file to the workstation).

```
cd <Disk name>:\install
```

For example:

```
cd D:\install
```

2. If SAP BusinessObjects Business Intelligence Platform has not been installed in its default folder, define the SAP BusinessObjects Business Intelligence Platform installation folder by entering the following command:

```
set BOBJ_InstallDir=<SAP BusinessObjects Business Intelligence Platform installation folder>
```

For example:

```
set BOBJ_InstallDir=C:\my directory\Business Objects
```

3. If you imported your reports into a folder whose name is not **Asset Manager**, define the folder where the reports were imported by entering the following command:

```
set AMRootReportsFolder=<reports import folder>
```

For example:

```
set AMRootReportsFolder=Asset Manager Reports
```

4. You are about to execute the **tool_step3_buildReportsList.bat** script to compile the list of Asset Manager reports loaded in SAP BusinessObjects Business Intelligence Platform. By default, the result of this script is stored in the **C:\am_reports.dat** file. If you would like to change the location or name of the result file, enter the following command:

```
set ReportsListFile=<path and file name>
```

For example:

```
set ReportsListFile=C:\temp\asset_manager_reports.dat
```

5. Execute the **tool_step3_buildReportsList.bat** script by entering the following command:

```
buildReportsList.bat
```

buildReportsList.bat generates a file and provides you with the name of the file when execution is complete.

Note: If you used manual installation and you defined the CMS port, you must modify the **CMSport** property in the resource.bat file in the lib folder.

6. To exclude reports from being imported into the Asset Manager database, open the file that was generated by **tool_step3_buildReportsList.bat** and put the **#** character at the beginning of each report line to be excluded.
7. At the DOS command prompt, enter the following command to define the source language of the data:

```
set BOBJ_ClientLanguage=<Asset Manager database language>
```

Select a language using one of the following two-letter language codes: **EN, DE, ES, FR, RU, IT, JP, KO, CHS, PT, NL**.

8. Start a Windows Explorer.
9. Change to the `\lib\reports` folder.
10. Copy the `am_reportsDSDef_xxx.dat` file, where **xxx** represents the SAP Crystal Reports installation language.
11. Paste it in any folder.
12. Edit the `am_reportsDSDef_xxx.dat` file.
13. Populate the parameter after **database** by specifying the name of the connection to the Asset Manager database (See "[Create a connection to the Asset Manager database](#)" on page 76). Do not modify any of the other parameters.
14. Return to the DOS command prompt window.
Reference the `am_reportsDSDef_xxx.dat` file that you just modified by entering the following command:

```
set DataSourceDefFile=<Full name of the modified am_reportsDSDef_xxx.dat file>
```

15. Enter the following command to execute the `tool_step4_changeReportsDS.bat` script:

```
tool_step4_changeReportsDS.bat
```

When you execute this script, you need to enter the password of the Crystal Report administrator.

Note: If you used manual installation and you defined the CMS port, you must modify the **CMSport** property in the `resource.bat` file in the `lib` folder.

16. You have now associated a custom source database to each Asset Manager report in SAP BusinessObjects Business Intelligence Platform.

Configuring SSO

Configuring single sign-on can avoid having to authenticate more than once (when connecting to the Asset Manager Web client, when accessing a report or when the report needs to retrieve data from the Asset Manager database).

Single sign-on lets users access all services and does not require them to re-enter authentication information multiple times.

1. Copy **am-reporting.common-96.jar** and **am-reporting.serverside-96.jar** (which can be found in the **lib\sso** folder) to the **lib** folder of the BusinessObjects Business Intelligence Platform for **Asset Manager** Web application. For example: **C:\Program Files (x86)\SAP BusinessObjects\tomcat\webapps\clientapi\WEB-INF\lib**.
2. Update the **web.xml** of **Asset Manager** Web tier (for example, C:\Tomcat\webapps\AssetManager\WEB-INF\web.xml). Search for the **SSO filter** section and make sure that the following two configurations exist, if not, you need to add them manually.

```
<!-- Define SSO Filter and associated mapping -->
  <filter>
    <filter-name>SSO Filter</filter-name>
    <filter-class>com.hp.sw.bto.reporting.sso.client.SSOClientFilter</filter-
class>
  </filter>
```

```
<!-- Crystal Reports Server SSO filter mapping -->
  <!-- MUST be declared after Acegi filter mapping -->
  <filter-mapping>
    <filter-name>SSO Filter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

3. Update **web.xml** of BusinessObjects Business Intelligence Platform (for example, C:\Program Files(x86)\SAP BusinessObjects\tomcat\webapps\clientapi\WEB-INF). Search for the **SSO filter** section and make sure that the following four configurations exist, if not, you need to add them manually.

```
<!-- Define SSO Filter and associated mapping -->
  <!-- Define Asset Manager SSO Filter and associated mapping -->
  <filter>
    <filter-name>HPE Asset Manager SSO Authentication Filter</filter-name>
    <filter-
class>com.hp.sw.bto.reporting.sso.server.AuthenticationFilter</filter-class>
  </filter>
```

```
<filter-mapping>
  <filter-name>HPE Asset Manager SSO Authentication Filter</filter-name>
  <url-pattern>/ErsViewerServlet</url-pattern>
</filter-mapping>
```

```

<!-- The mapping for the JSP servlet -->
  <servlet>
    <servlet-name>HPE Asset Manager SSO Authentication Servlet</servlet-
name>
    <servlet-
class>com.hp.sw.bto.reporting.sso.server.DbLogonViewerServlet</servlet-class>
</servlet>

    <servlet-mapping>
      <servlet-name>HPE Asset Manager SSO Authentication Servlet</servlet-
name>
      <url-pattern>/ErsViewerServlet/*</url-pattern>
    </servlet-mapping>

```

4. Copy the **reporting.properties** file (located in the **lib\ssso** folder) into the **WEB-INF\classes** folder of the BusinessObjects Business Intelligence Platform web application (for example: C:\Program Files (x86)\SAP BusinessObjects\tomcat\webapps\clientapi\WEB-INF\classes). You can edit this file and customize its parameters, including:

Parameter name	Default value	Parameter description
server.bo.shared_secret	N/A	You must copy and paste the value of the shared secret from SAP Central Management Console to this parameter.
server.bo.admin_account	Administrator	SAP Crystal Reports administrator account (only used to create a user on-the-fly).
server.bo.cms_port	7400	SAP BusinessObjects Business Intelligence Platform database port number.
server.bo.default_user	AssetManager	If this parameter is defined, the connection to the SAP BusinessObjects Business Intelligence Platform will be made using this user name and not the user name provided by the single sign-on mechanism. If the user name does not exist, it is created on-the-fly if the on_the_fly_creation parameter has been enabled. Otherwise the connection attempt fails.
server.bo.user.on_the_fly_creation	true	Authorize or deny on-the-fly creation of a user in SAP BusinessObjects Business Intelligence Platform (the user will or will not be created in the SAP BusinessObjects Business Intelligence Platform database during their initial connection if

Parameter name	Default value	Parameter description
		they does not yet exist in the database).
server.bo.user.can_change_password	true	SAP Crystal Reports properties for the user when the user is created on-the-fly.
server.bo.user.must_change_password	false	Idem
server.bo.user.password_expires	false	Idem
server.bo.user.is_named	false	Idem
server.bo.user.password	Welcome	Default user password for a user created on-the-fly.
server.bo.user.description	user auto-created by the Hp Ers Sandbox	Default user description for a user created on-the-fly.
#server.bo.cms_name	N/A	By default, this parameter does not take effect. If your BusinessObjectServer and CMS server are not on the same machine, you must remove the comment tag "#" and set its value to the CMS server. For example, server.bo.cms_name=cms.company.com.

5. Configure the password for single sign-on:
 - a. Launch the Central Management Console.
 - b. Select the **Enterprise** authentication type and log on with the **Administrator** account (the password was defined during the installation process).
 - c. Click **Authentication** under the **Manage** group.
 - d. Double-click the **Enterprise** record, select the **Trusted Authentication is enabled** option.
 - e. Click **New Shared Secret**.
 - f. Click **Download Shared Secret**, then open the `TrustedPrincipal.conf` file with a text editor.
 - g. Copy and paste the value of the shared secret to the **server.bo.shared_secret** property in the **reporting.properties** file.
 - h. Click **Update**.

Caution: The single sign-on mechanism will not work if the host used for the Asset Manager

application does not use the same DNS domain as the host for the SAP BusinessObjects Business Intelligence Platform. This is due to certain restrictions imposed by Internet standards such as cookie propagation.

If both hosts are located in two different DNS domains, users will need to authenticate at each connection: when connecting to Asset Manager, when connecting to SAP BusinessObjects Business Intelligence Platform and when connecting to the Asset Manager database.

Also, for single sign-on to operate correctly, you must enter the domain name in:

- The SAP BusinessObjects Business Intelligence Platform server name: "[Declaring the access URL to the SAP BusinessObjects Business Intelligence Platform](#)" below.
- The Asset Manager Web Tier server name: **http://<HPE Asset Manager Web Tier server name with domain name>:<Port used by HPE Asset Manager Web Tier>/AssetManager**

Declaring the access URL to the SAP BusinessObjects Business Intelligence Platform

1. Start the Asset Manager Windows or Web client.
2. Connect to the database as Admin.
3. Start the **Edit the URL address of the application servers...** wizard (BstBackEndOpt) (**Administration/ User actions/ Edit the URL address of the application servers...** link on the navigation bar).

4. Populate this wizard as follows:

Field	Value
Add and/or modify properties of the application servers page	
Name of the application server to add	If, in the lower part of the page, there is no line whose Application column equals SAP BusinessObjects Crystal Reports Server , populate this field with SAP BusinessObjects Crystal Reports Server , then click Add the application server .
List of applications, line whose Application column equals SAP BusinessObjects Crystal Reports Server, Server's URL address column	<p>http or https://<SAP BusinessObjects Business Intelligence Platform server name>:<Port used by SAP BusinessObjects Business Intelligence Platform></p> <p>Example:</p> <p>http://BusinessObjectServer:7080</p> <p>Caution: In order for single sign-on to operate correctly, you must enter the domain name in the SAP BusinessObjects Business Intelligence Platform server name.</p>
Summary of changes page	Verify the contents of the page.
Click Finish	
Windows client: Click OK	

5. Windows client: Reconnect to the database (**File/Connect to database** menu).

Mapping the report file name with Crystal report ID

1. Start the Asset Manager Windows or Web client.
2. Connect to the database as Admin.
3. Select the **Administration/ Business home page** navigation menu.
4. Select the domain you want to add crystal report to. For example, Portfolio.
5. Select the **Reports** tab.
6. Add the report you want display on the business home page, for example: Analysis of levels of consumable stocks.

7. Display the detail of each of your selected reports and update the **File** field value with crystal report ID (accessed from the **ID** value in the **Properties** page of the report's default settings in the BusinessObjects Business Intelligence Platform Central Management Console). For example, for Analysis of levels of consumable stocks, the Crystal report ID is 1200.

Configuring the sysCoreWebCrystal calculated field

To enable Asset Manager web client to access the reports, you must modify the **sysCoreWebCrystal** field.

To modify the **sysCoreWebCrystal** calculated field:

1. Start the Asset Manager Windows client.
2. Open the **calculated fields** screen (**Administration/ system/ Calculated fields...** navigation menu).
3. Select the **Display report** (sysCoreWebCrystal) calculated field.
4. Replace the BASIC syntax with the following scripts:

```
Dim strURL as String
Dim strServerCrystal as String
Dim strFileName as String
strFileName = [FileName]
strURL = ""
if IsNumeric(strFileName) then
strServerCrystal = amBackEndServerPath("SAP BusinessObjects Crystal Reports
Server")
if Len(strServerCrystal)>0 and Len(strFileName)>0 then
strURL = strServerCrystal
strURL = strURL + "/clientapi/ErsViewerServlet?reportId="
strURL = strURL + strFileName
end if
end if
RetVal = strURL
```

Migrating from BusinessObjects Enterprise XI 3.1 to BusinessObjects Business Intelligence Platform 4.1

You can migrate from your customized BusinessObjects Enterprise XI 3.1 system (including reports, user rights, and so on) to BusinessObjects Business Intelligence Platform 4.1, thereby enabling the Asset Manager 9.61 Windows and Web client to display the Crystal Reports correctly.

1. On the server that has BusinessObjects Business Intelligence Platform 4.1 installed, open the Upgrade management tool.
2. Click **Complete Upgrade**, click **Next**.
3. In the **Upgrade Scenario** drop-down list, select **Live to Live**.
4. In the **Source** section, enter the relevant information of the BusinessObjects Enterprise XI 3.1 server.
5. In the **Destination** section, enter the relevant information of the BusinessObjects Business Intelligence Platform 4.1 server.

Note: Use <hostname>:<port> as the CMS Name.

6. Click **Next**.
7. Click **Start**.

Display a report

This section includes:

To display a report	101
Available reports	103
How is the report displayed	103
How does Asset Manager generate the URL address of the reports	104

To display a report

How a report is displayed depends on the report type (detail, list or graph):

Detail reports

Caution: Detail reports are available via the Windows client, but not via the Web client.

A detail report can be displayed in various ways:

Displaying a report by selecting it in the navigation bar

1. Display the list containing the record that is the object of the report.
2. Select the record that is the object of the report.

Note: You can only select one record.

3. Expand the navigation bar link corresponding to the report's functional domain.

Tip: The report's functional domain is defined in the **Domain** (Domain) field of the report's detail.

4. Click the report name in the navigation bar.

Displaying a report from the list of reports (Windows client only)

1. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).
2. Display the report detail.
3. Click **Preview**.
4. Select the record that is the object of the report.
5. Click **OK**.

Displaying a report using the File/ Print menu (Windows client only)

1. Display the list of records that are the object of the report.
2. Select the records that are the object of the report.
3. Select the **File/Print** menu.
4. Populate the **Type** field with the value **Detail report (Crystal Reports)**.

5. Select the report in the **Report** list.
6. Click **Preview**.

List reports and graphs

A list or graph report can be displayed in various ways:

Displaying a report by selecting it in the navigation bar

Expand the navigation bar link corresponding to the report's functional domain.

Tip: The report's functional domain is defined in the **Domain** (Domain) field of the report's detail.

Click the report name in the navigation bar.




Displaying a report from the list of reports (Windows client only)

1. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).
2. Display the report detail.
3. Click **Preview**.

Displaying a report via the File/ Print menu (Windows client only)

1. Select the **File/Print** menu.
2. Populate the **Type** field with the value **Non-contextual report (Crystal Reports)**.
3. Select the report in the **Report** list.
4. Click **Preview**.

Displaying a report with automatic refresh (Windows client only)

1. Select the **Tools/ Reporting/ Crystal Reports statistics** menu.
2. Populate the **Nature** field if you want to filter the list of reports (list to the right of the **Nature** field).
3. Select the report in the list to the right of the **Nature** field.
4. To refresh a report: Click the  icon.
To set how often a report is automatically refreshed: right-click the  icon.
To modify the zoom factor (3 levels): Click the  icon.

Available reports

Below are the conditions that a report must meet before they can be used in the Windows and Web clients:

Condition	Windows client	Web client
Where the report is stored	<ul style="list-style-type: none"> Asset Manager database SAP BusinessObjects Business Intelligence Platform 	SAP BusinessObjects Business Intelligence Platform
	<p>Caution: The Web and Windows clients consider that a report is available via the SAP BusinessObjects Business Intelligence Platform if the BusinessObjects Business Intelligence Platform was declared via the Edit the URL address of the application servers... wizard (BstBackEndOpt) and if the report's File field (database) contains a numerical value.</p>	
Report type	<ul style="list-style-type: none"> Detail report <p>Caution: A valid context must be selected to have the report appear in the navigation bar.</p> <ul style="list-style-type: none"> List report Graph 	<ul style="list-style-type: none"> List report Graph
Visible in menus (bVisible) checkbox	Checked	Checked

How is the report displayed

Windows client:

- If the **.rpt** file is stored in the Asset Manager database: The report is displayed by the Windows client.
- If the **.rpt** file is not stored in the Asset Manager database, but is stored on the BusinessObjects Business Intelligence Platform: The report is displayed by an Internet browser.

Web client: The report is displayed by the Web client.

How does Asset Manager generate the URL address of the reports

Asset Manager uses the **Display report** (sysCoreWebCrystal) calculated field to generate an HTML **<A>** anchor with an **HREF** attribute whose label is the report's label. It points to a URL address generated by concatenating:

- The URL address of the SAP BusinessObjects Business Intelligence Platform defined by the **Edit the URL address of the application servers...** wizard (BstBackEndOpt)
["Declaring the access URL to the SAP BusinessObjects Business Intelligence Platform" on page 97.](#)
- A text string specified in the calculated field's script
["Configuring the sysCoreWebCrystal calculated field" on page 99.](#)
- The report's **File** (FileName) field
["Mapping the report file name with Crystal report ID" on page 98.](#)

When you click the link named after the report, Asset Manager processes the URL address.

Note: By default, the **Display report** (sysCoreWebCrystal) calculated field is configured to access SAP BusinessObjects Business Intelligence Platform version 4.1.

Making reports available

Before the report can be used by the Windows and/or Web clients, an **.rpt** file of the report must be created and the report must be stored in one of the following databases:

- Asset Manager database
- SAP BusinessObjects Business Intelligence Platform

To determine in which database the Crystal Reports needs to be stored, see [Where are the reports stored?](#)

This section includes:

Obtaining .rpt files	105
----------------------------	-----

Store reports in the Asset Manager database	105
Storing reports in the SAP BusinessObjects Business Intelligence Platform	107

Obtaining **.rpt** files

The Crystal Reports come from several sources:

- Reports that you create yourself using BusinessObjects Business Intelligence Platform.
- Reports already stored in the Asset Manager database.
If you want to transfer them to the BusinessObjects Business Intelligence Platform database, you must export them one by one using the Windows client by clicking the **Export** button on the report's detail page.
- Asset Manager line-of-business data reports.

These reports are located in the **datakit** sub-folder of the Asset Manager installation folder.

Note: The list of available reports is available in the **reports.txt** file of the **datakit\standard\reports** sub-folder, located in the Asset Manager installation folder.

The Crystal Reports are stored in the **\datakit\standard\reports\rpt** folder.

Do not modify the structure of the **Reports.txt** file since the import script of the reports uses it. On the other hand, if you only want to import a selection of reports into your working database, you can delete **full** lines from this file before executing the import script. You may also add your own reports.

Store reports in the Asset Manager database

Reports stored in this manner can only be accessed via the Windows client.

Importing reports one by one into an existing Asset Manager database

To insert the reports one by one:

1. Start the Asset Manager Windows client.
2. Connect to the database.
3. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).

4. Create a new report.
5. Click **Import**.
6. Select the **.rpt** extension file that corresponds to your needs in the **\\datakit\standard\reports\rpt** sub-folder of the Asset Manager installation folder.

Importing all reports supplied with Asset Manager when creating a new database

To import reports during the database creation:

1. Launch Asset Manager Application Designer.
2. Select the **File/ Open** menu.
3. Select the **Open database description file - create new database** option.
4. Select the **gbbase.xml** file, located in the **config** sub-folder of the Asset Manager installation folder.
5. Start the database creation wizard (**Action/ Create database** menu).
6. Follow the instructions in the **Administration** guide, chapter **Creating, modifying and deleting an Asset Manager database**.
In the page **Data to import**, select **Crystal Reports**.

Importing all reports supplied with Asset Manager into an existing database

To import reports into an existing database:

1. Start Asset Manager Application Designer.
2. Select the **File/ Open** menu.
3. Select the **Open database description file - create new database** option.
4. Select the **gbbase.xml** file, located in the **config** sub-folder of the Asset Manager installation folder.
5. Start the database creation wizard (**Action/ Create database** menu).
6. Populate the pages of the wizard as follows (navigate through the wizard pages using the **Next** and **Previous** buttons):

Generate SQL script / Create database page

Fields	Value
Database	Select the connection to the database into which you wish to import the reports.
Creation	Import line-of-business data.
Use advanced creation options	Do not select this option

Creation parameters page

Fields	Value
Password	<p>Enter the administrator's password.</p> <p>Note: The Asset Manager database administrator is the record in the Employees and departments (amEmplDept) table for which the Name (Name) field is set to Admin.</p> <p>The database connection login is stored in the User name (UserLogin) field. The administration name is Admin.</p> <p>The password is stored in the Password field (LoginPassword).</p>

Data to import page

Fields	Value
Available data	Select the option Crystal Reports .
Stop import if error	Select this option for the import to stop if a problem is encountered.
Log file	Full name of the file to which all import operations, including errors and warnings, are logged.

- Execute the options defined using the wizard (**Finish** button).

Storing reports in the SAP BusinessObjects Business Intelligence Platform

Reports stored in this manner can be accessed via the Windows and the Web clients.

For each report that you want to make available to users:

1. Start the SAP BusinessObjects BI Platform Central Management Console.
2. Start the Asset Manager Windows or Web client.
3. Connect to the database.
4. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).
5. Create the report by populating the following fields:

Name of the field or link	SQL name of the field or link	Comment
Label	Title	Name of the report in the BusinessObjects Business Intelligence Platform Note: This field is not required but can prove useful.
File	FileName	Identifier of the report in the BusinessObjects Business Intelligence Platform

How to modify a Seagate Crystal report

This section includes:

Reports stored in the Asset Manager database	108
Reports stored in the SAP BusinessObjects Business Intelligence Platform	109

Reports stored in the Asset Manager database

1. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).
2. Display the report detail.
3. Export the report (**Export** button) to create a **.rpt** file.
4. Modify the **.rpt** report using SAP Crystal Reports 2013 and save it.
5. Display the report detail again.
6. Import the **.rpt** file that has been modified (**Import** button).
7. Save the modifications (**Modify** button).

Reports stored in the SAP BusinessObjects Business Intelligence Platform

Modify the report directly with BusinessObjects Business Intelligence Platform without modifying its identifier in the SAP BusinessObjects Business Intelligence Platform (see ["Storing reports in the SAP BusinessObjects Business Intelligence Platform" on page 107](#)).

Creating a detail report

A "detail report" is a report that prints the detail information on one or more records selected in a list.

This section includes:

Example of utilization	109
Configuring reports under SAP Crystal Reports	109

Example of utilization

1. Display the list of assets.
2. Select an asset.
3. Select the **File/ Print** menu item.
4. Set the **Type** field to "Detail report (Crystal Reports)".
5. Select the report.
6. Print.

This procedure prints a detail report for each selected record.

Configuring reports under SAP Crystal Reports

To obtain a detail report, follow the procedure below (example taken under SAP Crystal Reports 2013):

1. Use the **Report/ Formula Workshop-> New** menu item to create a formula field. Its name must respect the following syntax:

```
<SQL name of the table for which the report is contextual>Id
```

Note: You must respect the case of the SQL names of tables.

For example, to create a contextual report on the table of assets, the formula is:

```
amAssetId
```

Note: Do not confuse the syntax of the formula field name with the SQL name of the primary key field. For example, the primary key of the table of assets is "IAstId", which is different from "amAssetId".

The "CurrentUserId" formula (respect the case) makes it possible to identify the user printing the report. When printed, this formula takes the value of the identifier number (that is, the value of the field with SQL name: "IEmplDeptId" for the current login) of the user connected to the Asset Manager database.

If you want to see the result of the report for a given record in the contextual table, edit the formula field and enter the primary key of the table for an existing record in the Asset Manager database.

For example:

```
512
```

Note: Edit the formula field in the window, which is automatically displayed when you confirm the name of the new formula field. If the formula field exists already, click the **Edit** button to edit it.

2. Use the **Report/ Edit Selection Formula/ Record** menu item to edit the selection formula. It uses the following syntax:

```
{<SQL name of the context's table>.<SQL name of the field that is a primary key>} = @<Name of the formula's field>}
```

The case used for the SQL names of tables and fields is unimportant. Example:

```
{amAsset.IAstId} = {@amAssetId}
```

Using the above procedure, Asset Manager automatically identifies the report as being contextual when it is imported into the database. You will see this when you perform the following:

1. Access the list of reports using the **Tools/ Reporting/ Reports** menu item.
2. Create a new report.

3. Import the Crystal Reports file (.rpt extension) by clicking the **Import** button.
4. Once this file has been added, you will notice that the **Table** field (SQL name: TableName) shows the SQL name of the context table. If this is not the case, verify the formula field and the selection formula in the SAP Crystal Report.

Printing a report

How a report is printed depends on the report type (detail, list or graph).

This section includes:

Detail reports (Windows client only)	111
List reports and graphs	112

Detail reports (Windows client only)

Caution: Detail reports are available via the Windows client, but not via the Web client.

Various ways to print a detail report exist:

Select the report in the navigation bar

1. Display the list containing the record that is the object of the report.
2. Select the record that is the object of the report.

Note: You can only select one record.

3. Expand the navigation bar link corresponding to the report's functional domain.

Tip: The report's functional domain is defined in the **Domain** (Domain) field of the report's detail.

4. Click the report name in the navigation bar.
5. Click the **Print** icon.

From the list of reports

1. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).
2. Display the report detail.
3. Click **Preview**.
4. Select the record that is the object of the report.
5. Click **OK**.
6. Click the **Print** icon.

Via the **File/ Print** menu.

1. Display the list of records that are the object of the report.
2. Select the records that are the object of the report.
3. Select the **File/Print** menu.
4. Populate the **Type** field with the value **Detail report (Crystal Reports)**.
5. Select the report in the **Report** list.
6. Click **Print**.

List reports and graphs

A list or graph report can be printed in various ways:

Select the report in the navigation bar

1. Expand the navigation bar link corresponding to the report's functional domain.
Tip: The report's functional domain is defined in the **Domain** (Domain) field of the report's detail.
2. Click the report name in the navigation bar.
3. Click the **Print** icon.

From the list of reports (Windows client only)

1. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).
2. Display the report detail.
3. Click **Preview**.
4. Click **OK**.
5. Click the **Print** icon.

Via the **File/ Print** menu (Windows client only)

1. Select the **File/Print** menu.
2. Populate the **Type** field with the value **Non-contextual report (Crystal Reports)**.
3. Select the report in the **Report** list.
4. Click **Print**.

Identifying Crystal Reports specific to a given module

To identify the Crystal Reports specific to a given module:

1. Start the Asset Manager Windows client.
2. Display the reports (**Administration/ Reporting/ Reports** link on the navigation bar).
3. Right-click in the list of the window that opens.
4. Select **Utilities/ Configure list** from the shortcut menu.
5. Add the **Domain** link (Domain) to the columns in the list.
6. Click **OK**.
7. Sort the list on the **Domain** column.
8. The reports for a module are identified by the domain name.
Example: **/Asset lifecycle/Software Asset Management/Reports/**

Associating a report with a button in a screen (Windows client) or with actions in a screen (Web client)

A report can be associated with a button in a screen (Windows client) or with actions in a screen (Web client) in several different ways.

For more information, see:

- The **Tailoring** guide, **Customizing the database** chapter, **Customizing existing objects/ Customizing objects / Customizing a screen/ Buttons** section.
- The **Tailoring** guide, chapter **Customizing the database**, section **Creating new objects/ Creating action buttons**.

Limitations

- How SAP Crystal Reports are used depends on whether you are accessing them via the Web or Windows client, and whether or not you are using reports stored in the Asset Manager database or the SAP BusinessObjects Business Intelligence Platform.

This type of limitations are described in the **Web Implementation** guide, chapter **Differences between the Windows client and the Web client**, section **Reporting**.

- Dynamic parameters are not supported by Asset Manager. When running a crystal report with a dynamic parameter, Asset Manager may stop responding.

Note: Dynamic parameters are used to interactively select the data from the database when the report is running.

Chapter 5: Dashboards

Asset Manager allows you to create dashboards that include reports and statistics. A dashboard offers an executive view of information to facilitate decision-making. Just like a car or airplane dashboard, it is meant to include relevant and critical data to help you anticipate problems and help the smooth running of your company. A dashboard usefulness is thus directly linked to the choice of indicators that are included.

This chapter includes:

Creating a dashboard	115
Dashboard example	116

Creating a dashboard

A dashboard is defined by a **Dashboard** type action.

To create a dashboard:

1. Select the **Administration/ Actions** link on the navigation bar and click **New** in the action detail screen.
2. Choose a name for the dashboard and select **Dashboard** as the action **Type**.
3. If you want the dashboard to appear in the navigation bar's tree structure, define a **Domain** for the dashboard.

Caution: If no **Domain** is specified for the dashboard, you will not be able to display it in the Web client.


You can also specify other properties that are common to all actions.


Note: For further information, refer to the [Actions](#) chapter of this manual.

4. From the **Indicators** tab, select the indicators that you want to add to the dashboard. The indicator association information is stored in the **Dashboard item** (amDashboardItem) table. Two type of objects are used as indicators for dashboards in Asset Manager: reports and statistics. This section does not discuss creating statistics or reports: Creating SAP Crystal Reports is detailed in chapter [SAP Crystal Reports](#) of this guide and statistics in chapter

Statistics.

To associate an indicator:

- a. Select the **Indicators** tab in the dashboard detail and click  (Windows client) or **Add** (Web client).
- b. Choose a **Name** to identify the indicator within the dashboard.
- c. Define the **Type** of the indicator (report or statistic).
- d. Finally, select a **Report** or a **Statistic**.

Note: Click  (Windows client) or **Delete** (Web client) to remove an indicator from the dashboard. A message is displayed and explains that the indicator association with the dashboard will be deleted. This means that the object stored in the **Dashboard item** (amDashboardItem) table will be deleted.

5. You may also check the **Default refresh** checkbox. When this checkbox is checked, data from the dashboard is refreshed according to the periodicity defined by the **Statistics/ Refresh frequency of statistics**.

Note: The **Statistics/ Refresh frequency of statistics** option is available via the **Edit/ Options** menu in the Windows client.

If this checkbox is not checked the dashboard is static and the data is not dynamically refreshed.

Dashboard example

The following example describes the creation of a dashboard used by the Human Resources of a company. The following indicators have been defined as relevant for the dashboard:

- Breakdown of Men / Women in the company
- Breakdown of employees by department and site
- Breakdown of employees by time in company

These indicators were created in the [Non-scripted statistics](#) section of chapter [Statistics](#). Refer to that section to create the statistics required in this example.

Step 1: Create the dashboard




1. Select the **Administration/ Actions** navigation menu and click **New** to create a new action.
2. Populate the fields of the new record with the following data:

Field	Value
Name	HR Dashboard
Type	Dashboard

3. Validate your creation by clicking **Create** (Windows client) or **Save** (Web client).

Step 2: Associating indicators

Select the **Indicators** tab in the detail of the newly created dashboard. The various statistic indicators will be successively added according to the following procedure:

1. Click  (Windows client) or **Add** (Web client).
2. Populate the **Name** field of the indicator. You can use the names of the statistics that you want to add.
3. Select **Statistic** as the indicator **Type**.
4. Click on the  (Windows client) or  (Web client) selection list next to the **Statistic** field, then select the statistic that you want to add.
5. Click **Add** (Windows client: or the **Add** button with twin arrows if you want to add several indicators).

If you are using the Windows client, click **Modify**.

Step 3: Open the dashboard and navigate in the data

Launch the dashboard either by using the **Tools/ Actions** menu or by clicking its name in the Windows or Web client navigation bar where you defined it in the **Domain** (Domain) field.

Dashboard statistics can be customized once they are displayed by using the options of the contextual menu. You can also click on the data zones to display the detail of a statistical sample. For example, if you click on the zone of the 3D sector that represents the male population Asset Manager opens the departments and employees screen and filters the data on the **Mr/Mrs** (MrMrs) field, using **Mr** as the value for the filter.

Note: Navigation is not available for the reports.

Chapter 6: Statistics

Statistics are used to display database information in charts.

Statistics cannot be displayed on their own and must be displayed via a dashboard or a business home page. See [Making the statistics viewable](#)

Do not confuse **statistics** as described in this chapter with **SAP Crystal Reports statistics** described in chapter [Displaying a report with automatic refresh \(Windows client only\)](#):

- SAP Crystal Reports statistics are used to display a SAP Crystal Reports report that can be updated automatically.
- When you display a **statistics** chart, you can zoom in on items of the chart to display the records that are represented by the item. Zooming is not possible with **SAP Crystal Reports statistics**.

This chapter includes:

Create a statistic	119
Using a script instead of a query	121
Optimizing performance	122
Zooming in on statistics details	123
Defining the maximum number of groups to represent individually	123
Making the statistics viewable	124
Displaying a statistic	125
Examples of statistics	125
Defining custom colors for charted statistics	139

Create a statistic

To create a statistic:

1. Display the statistics (**Administration/ Reporting/ Statistics** link on the navigation bar).
2. Create a new statistic (**New** button) and populate the various fields.
3. If you want to store the results of the statistic in the database, enable the **Store results** (bStored)

option.

See [Optimizing performance](#).

4. You can define statistics in one of two ways:

- By using a standard **query**:
 - i. Leave the **Script** (bScript) box unchecked.
 - ii. On the **Query** tab, enter the different parameters to create your statistic:
 - Required components:
 - Specify the table with the records to analyze in the **Table** (TableName) field.
 - To specify the field or link to use to group the elements, use the **Group by** (GroupBy) field.
 - Optional components:
 - If you want to use only some of the records in the table to create the statistic, in the **Filter** field, specify the filter to use to return the records to be included in the statistic.
If this field is empty, all of the records are taken into account.
 - If you want to group by a second value, once the **Group by** field has been populated, you can use the **Sub-group by** (SubGroupBy) field to group by this second value.
 - If you want to use an expression to generate chart values, check the **Use an expression** (bUseAgregExpr) box.
Next, populate the **Expression** (AqlAgregExpr) field.
This lets you perform operations (for example, averages) or lets you account for batches.
- By writing a complex **script** to display statistics
 - i. Check the **Script** (bScript) check box.
 - ii. On the **Script** tab, enter the script used to calculate the statistics data.

Read the [Using a script instead of a query](#) paragraph to learn about the differences between scripts and queries.

Read the [Scripted statistics](#) paragraph for examples of scripted statistics.

Caution: If you do not specify a **Domain** for the statistic, it will not be available via the Web client (even after it is associated with a dashboard or a business home page. See [Making the](#)

statistics viewable).

Caution: Web client: If the selected representation type (seType field) is **Curves**, the statistic can only be displayed if the displayed values are **numbers**. The **Group by** (GroupBy) and, possibly, the **Sub-group by** (SubGroupBy) fields must point to numerical values.

5. After populating the fields, click **Create** (Windows client) or **Save** (Web client).

If you are using the Windows client, you can preview the chart by:

1. Click **Test**.
2. Press **ESC** to close the chart.

Using a script instead of a query

As explained in paragraph [Create a statistic](#), there are two ways to describe statistics:

- Via a simple query.
- Via a complex script.

This section includes:

Why use a script instead of a query?	121
Syntax of scripts	122

Why use a script instead of a query?

Standard queries are used to display table data:

- Using one or two grouping criteria,
- On database fields and certain calculated fields,
- Possibly using a filter,
- And, if required, a calculation method (expression) is applied.

If you want to define a complex statistic, you will need to define it using a **script**.

For example, if you want to group records by period, compare records in different tables or group using data from a **full name** type field, you will need to write a **script**.

Syntax of scripts

Review:

- the examples in the [Scripted statistics](#) section to see the syntax that is used when writing scripts.
- [Defining custom colors for charted statistics](#)

Optimizing performance

This section includes:

What can cause performance issues?	122
Solution to work around these performance issues	123
How do I implement this solution?	123

What can cause performance issues?

Statistics are displayed via dashboards.

When a dashboard is displayed, Asset Manager accesses the appropriate tables via one or more statistics.

This can cause the following:

- Slows down the client station.
- Slows down access to certain tables for other users.
- Slows down the display of results as they are calculated in real time. Also, more than one statistic may need to be calculated for a given dashboard.

Solution to work around these performance issues

To help reduce performance issues, you can generate statistics automatically and on a regular basis, and have the results stored in the **Statistics memos** (amStatMemo) table.

In this case, when a dashboard is displayed, Asset Manager checks this table and not the actual table from which the data would otherwise be collected in real time. This greatly reduces calculation times.

The advantages of this option are:

- It does not penalize table access for other users submitting queries.
- Queries are launched by Asset Manager Automated Process Manager and not by user stations.

How do I implement this solution?

To implement this solution, use the **Update statistics results (sysStatsUpdate)** workflow.

Note: By default, the group for this workflow is empty, you must assign a group for this workflow.

Zooming in on statistics details

When you display the chart, you can access the details for each zone by clicking the zone. This opens the statistic's table filtered using the group by and filter criteria.

Defining the maximum number of groups to represent individually

Note: This functionality is only available in the Windows client.

We suggest that you not use more than twenty value groups to keep the chart legible.

The remaining groups can be grouped into a **Other values** group.

By default, and to keep the charts legible, the number of groups to be represented individually should not be more than 20. In this case, the first 20 value groups will be represented and the others, if any, will be grouped into a group named **Other values**.

The value groups displayed on the chart are sorted by value (except for the **Other values** group which is placed last, and which includes all groups exceeding the maximum number of groups to represent).

To modify the maximum number of groups to represent:

1. Open the options groups (**Edit/ Options** menu).
2. Expand the **Statistics/ Maximum number of values to take into consideration** branch.
3. Define the number of groups to represent individually.

Making the statistics viewable

This section includes:

From dashboards	124
From business home pages	125


From dashboards

Before a statistic can be displayed, you can associate it with a dashboard as follows:

1. Display the actions (**Administration/ Actions** link on the navigation bar).
2. Create a new action (**New** button).
3. Make sure you populate the **Type** (seActionType) field by selecting the **Dashboard** value and the **Domain** (Domain) field to specify the navigation bar node on which the dashboard will be available.

Caution: If you do not enter a value in the **Domain** field, the dashboard will not be visible in the Web client.

4. Validate the creation by clicking **Create** (Windows client) or **Save** (Web client).
5. On the **Indicators** tab, create the statistic indicators according to the following procedure:

- a. Click  (Windows client) or **Add** (Web client).
- b. Populate the **Name** (Name) field of the indicator. You can use the names of the statistics that you want to insert.
- c. In the **Type** (seType) field, select **Statistic**.
- d. Next, populate the **Statistic** link.


Perform these steps as many times as there are statistics to add.

6. If you are using the Windows client, update the dashboard (**Modify** button).
Your dashboard has now been created and can be used.

To use these statistics in your dashboards, read the following chapter: [Dashboards](#).

From business home pages

In order to use the statistics, you can associate them with a business home page as follows:

1. Create a business home page (**Administration/ Business home pages** link on the navigation bar).
2. On the **Statistics** tab, insert the desired statistics by clicking the  icon (Windows client) or the **Add** button (Web client), and then select the statistics.

To learn more about business home pages, read the **Tailoring** guide, section **Customizing Web clients**, chapter **Business home pages**.

Displaying a statistic

Statistics cannot be displayed on their own and must be displayed via a dashboard.

Refer to chapter [Dashboards](#) in this guide to see how statistics are used in a dashboard.

Examples of statistics

In its line-of-business data, Asset Manager provides examples of statistics that are generated in the database via dashboards.

These dashboards are available in the different line-of-business data.

They are available by default in the demonstration database.

However, to have them in your production database, you must import the line-of-business data of which they are a part.

An example of importing line-of-business data is provided in the **Procurement** guide, chapter **General overview**, section **Preliminary steps**, paragraph **Importing the Line-of-business data into an existing database**.

Non-scripted statistics

Example 1: Statistic on the average price of computers by cost center and CPU frequency

We want to view the statistic for the average price of computers by cost center and CPU frequency. This statistic:

- Presents two groups: Cost center and PC frequency
- A filter used for the computers
- Uses an expression to calculate the average price

To create this statistic:

1. Display the statistics (**Administration/ Reporting/ Statistics** link on the navigation bar).
2. Create a new statistic (**New** button).
3. Populate the fields and links shown in the following table:

Field or link (label)	Field or link (SQL name)	Value
Name	Name	Average price of computers by cost center and CPU frequency
General tab		
Representation type	seType	Horizontal bars
Key	bLegend	Check this box
Query tab		

Field or link (label)	Field or link (SQL name)	Value
Table	TableName	Computers (amComputer)
Group by	GroupBy	Portfolio.CostCenter.Title
If you entered the value for the previous field instead of selecting it from the list, you must validate your input or move the cursor to another field in order for the next field to be displayed.		
Sub-group by	SubGroupBy	ICPUSpeedMHz
Use an expression	bUseAgregExpr	Check this box
Expression	AqlAgregExpr	Avg(Portfolio.mAvgPrice)

4. Confirm your creation of the statistic by clicking **Create** (Windows client) or **Save** (Web client).
5. If you are using the Windows client, you can preview the chart by:
 - a. Click **Test**.
 - b. Press **ESC** to close the chart.

Example 2: Breakdown of Men / Women

We want to see the number of people working in the company broken down by men and women. You only need to use one criterion (the employee's title) to perform grouping. However, a filter used to set aside the services defined in the same table needs to be created so that information is not used in the statistic.

Start by creating this filter:

1. Display the queries (**Administration/ Queries** link on the navigation bar).
2. Create a new filter by populating the fields and links specified in the following table:

Field or link (label)	Field or link (SQL name)	Value
Name	Name	Employees only
Table	TableName	Employees and departments (amEmplDept)
Queries tab		
Query	memQueryText	bDepartment = 0

3. Confirm your creation of the filter by clicking **Create** (Windows client) or **Save** (Web client).
4. If you are using the Windows client, close all windows after you finish.

Now, create the statistic:

1. Display the statistics (**Administration/ Reporting/ Statistics** link on the navigation bar).
2. Create a new statistic (**New** button).
3. Populate the fields and links shown in the following table:

Field or link (label)	Field or link (SQL name)	Value
Name	Name	Breakdown Men Women
General tab		
Key	bLegend	Check this box
Representation type	seType	Sectors
3D Display	b3D	Check this box
Query tab		
Table	TableName	Employees and departments (amEmplDept)
Filter	Filter	Employees only
Group by	GroupBy	MrMrs

4. Confirm your creation of the statistic by clicking **Create** (Windows client) or **Save** (Web client).
5. If you are using the Windows client, you can preview the chart by:
 - a. Click **Test**.
 - b. Press **ESC** to close the chart.
6. Close all windows after you finish.

Example 3: Breakdown of employees by department and site

In this example, we want to see the breakdown of employees by department and site. The information is grouped using two criteria (the department and site) and a filter is applied for employees only (as previously).

1. Display the statistics (**Administration/ Reporting/ Statistics** link on the navigation bar).
2. Create a new statistic (**New** button).
3. Populate the fields and links shown in the following table:

Field or link (label)	Field or link (SQL name)	Value
Name	Name	Breakdown of employees by location
General tab		
Representation type	seType	Horizontal bars
Key	bLegend	Check this box
Query tab		
Table	TableName	Employees and departments (amEmplDept)
Filter	Filter	Employees only
Group by	GroupBy	Parent.Name
If you entered the value for the previous field instead of selecting it from the list, you must validate your input or move the cursor to another field in order for the next field to be displayed.		
Sub-group by	SubGroupBy	Location.FullName
Go back to the General tab		
Stacked representation	bStacked	Check this box

4. Confirm your creation of the statistic by clicking **Create** (Windows client) or **Save** (Web client).
5. If you are using the Windows client, you can preview the chart by:
 - a. Click **Test**.
 - b. Press **ESC** to close the chart.
6. Close all windows after you finish.

Example 4: Breakdown of employees by time in company

In this example, we want the data to be broken down by seniority. As previously, employees are filtered and grouped by seniority if necessary. Seniority is not a value stored in the database. But, the **Department and employees** table has a field that specifies when the employee began working at the

company (**Hire date**). This value can be used to calculate the employee's seniority. This is done using a calculated field whose value is then used to group the information.

To create the calculated field:

1. Display the calculated fields (**Administration/ System/ Calculated fields** link on the navigation bar).
2. Create a new record with the following information:

Field or link	Value
Name	Employee time in the company
Table	Employees and departments (amEmplDept)
Field type	AQL
Result type	Numeric
AQL syntax	<p>Round((DaysDiff(GetDate(), dHire) / 365), 0)</p> <p>The AQL query makes use of the <code>DaysDiff()</code> function that compute the duration between:</p> <ul style="list-style-type: none"> ○ the computation time: <code>GetDate()</code> function ○ the hire date: value of the Hire date (dHire) field <p>The duration is returned in days. The result is divided by 365, which corresponds to the number of days in a year. The result is then rounded down to the nearest whole number using the <code>Round()</code> function.</p>

3. Validate the creation by clicking **Create** (Windows client) or **Save** (Web client).
4. Close all windows.

Now, create the statistic:

1. Display the statistics (**Administration/ Reporting/ Statistics** link on the navigation bar).
2. Create a new statistic (**New** button).
3. Populate the fields and links shown in the following table:

Field or link (label)	Field or link (SQL name)	Value
Name	Name	Breakdown of employees by time in company
General tab		
Key	bLegend	Check this box
Representation type	seType	Curves
Query tab		
Table	TableName	Employees and departments (amEmplDept)
Filter	Filter	Employees only
Group by	GroupBy	Select the Seniority calculated field, which was previously created

4. Validate the creation by clicking **Create** (Windows client) or **Save** (Web client).
5. If you are using the Windows client, you can preview the chart by:
 - a. Click **Test**.
 - b. Press **ESC** to close the chart.
6. Close all windows after you finish.

Scripted statistics

Below are some examples of scripted statistics that you can test or modify to create your own.

You will find other scripted statistics examples in the demonstration database (select the **Administration/ Reporting/ Statistics** link on the navigation bar).

Example 1 (basic example)

The goal of this example is to demonstrate how to create a simple statistic (no data is retrieved from the database) and how to manage the information rendering.

In this example, three groups are created (**France**, **Germany** and **England**). Each group has a certain number of employees (**10**, **20** and **18**) and we want the chart to display the number of employees per country.

The following query needs to be generated:

```
<group Name="France" filtertable="amEmplDept" filter="Location.FullName like
'/France/%'" value="10"/>
<group Name="Germany" filtertable="amEmplDept" filter="Location.FullName like
'/Germany/%'" value="20"/>
<group Name="England" filtertable="amEmplDept" filter="Location.FullName like
'/England/%'" value="18"/>
```

To generate this query, the **Script** option (check its checkbox) will be used. This option lets you write a script which generates the statistical data. The **XmlAttribute** API will be used to convert predefined XML entities (See the **Programmer's Reference** for more details).

1. Display the statistics (**Administration/ Reporting/ Statistics** link on the navigation bar).
2. Create a new statistic (**New** button).
3. Populate the fields and links shown in the following table:

Field or link (label)	Field or link (SQL name)	Value
Name	Name	Number of employees per country
Script	bScript	Check this box
General tab		
Representation type	seType	Vertical bars
Key	bLegend	Check this box
Script tab		
Data script	Script	RetVal = "<group " & XmlAttribute("Name", "France") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/France/%'") & " " & XmlAttribute("value", "10") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<group " & XmlAttribute("Name", "Germany") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/Germany/%'") & " " & XmlAttribute("value", "20") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<group " & XmlAttribute("Name", "England") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/England/%'") & " "

Field or link (label)	Field or link (SQL name)	Value
		" & XmlAttribute("value", "18") & " />" & Chr(13) & Chr(10)

4. Validate the creation by clicking **Create** (Windows client) or **Save** (Web client).
5. If you are using the Windows client, you can preview the chart by:
 - a. Click **Test**.
 - b. Press **ESC** to close the chart.

Example 2 (with sub-group)

A sub-group has been added to the previous example and corresponds to the title used for each employee (**Mr**, **Mrs** or **Miss**).

Now, we'd like to view the chart representing the number of employees by title and country.

The following query needs to be generated:

```
<group Name="Mr.">
  <subgroup Name="France" filtertable="amEmplDept" filter="Location.FullName like
  '/France/%' and MrMrs='Mr'" value="5"/>
  <subgroup Name="Germany" filtertable="amEmplDept" filter="Location.FullName like
  '/Germany/%' and MrMrs='Mr'" value="9"/>
  <subgroup Name="England" filtertable="amEmplDept" filter="Location.FullName like
  '/England/%' and MrMrs='Mr'" value="10"/>
</group>
<group Name="Mrs.">
  <subgroup Name="France" filtertable="amEmplDept" filter="Location.FullName like
  '/France/%' and MrMrs='Mrs'" value="3"/>
  <subgroup Name="Germany" filtertable="amEmplDept" filter="Location.FullName like
  '/Germany/%' and MrMrs='Mrs'" value="7"/>
  <subgroup Name="England" filtertable="amEmplDept" filter="Location.FullName like
  '/England/%' and MrMrs='Mrs'" value="7"/>
</group>
<group Name="Miss">
  <subgroup Name="France" filtertable="amEmplDept" filter="Location.FullName like
  '/France/%' and MrMrs='Miss'" value="2"/>
  <subgroup Name="Germany" filtertable="amEmplDept" filter="Location.FullName like
  '/Germany/%' and MrMrs='Miss'" value="4"/>
  <subgroup Name="England" filtertable="amEmplDept" filter="Location.FullName like
```

```
'/England/%' and MrMrs='Miss'" value="1"/>
</group>
```

As previously, we will use the **Script** option to code this information.

1. Display the statistics (**Administration/ Reporting/ Statistics** link on the navigation bar).
2. Create a new statistic (**New** button).
3. Populate the fields and links shown in the following table:

Field or link (label)	Field or link (SQL name)	Value
Name	Name	Number of employees by title and country
Script	bScript	Check this box
General tab		
Key	bLegend	Check this box
Representation type	seType	Vertical bars
Script tab		
Data script	Script	<pre>RetVal = "<group " & XmlAttribute("Name", "Mr.") & " >" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "France") & " " & XmlAttribute ("filtertable", "amEmplDept") & " " & XmlAttribute ("filter", "Location.FullName like '/France/%' and MrMrs like '/Mr/%' ") & " " & XmlAttribute("value", "5") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "Germany") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/Germany/%' and MrMrs like '/Mr/%' ") & " " & XmlAttribute("value", "9") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "England") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/England/%' and MrMrs like '/Mr/%' ") & " " & XmlAttribute("value", "10") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "</group>" RetVal = RetVal & "<group " & XmlAttribute("Name",</pre>

Field or link (label)	Field or link (SQL name)	Value
		<pre> "Mrs.") & " >" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "France") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/France/%' and MrMrs like '/Mrs/%' ") & " " & XmlAttribute("value", "3") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "Germany") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/Germany/%' and MrMrs like '/Mrs/%' ") & " " & XmlAttribute("value", "7") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "England") & " " & XmlAttribute ("filtertable", "amEmplDept") & " " & XmlAttribute ("filter", "Location.FullName like '/England/%' and MrMrs like '/Mrs/%' ") & " " & XmlAttribute("value", "7") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "</group>" RetVal = RetVal & "<group " & XmlAttribute("Name", "Miss") & " >" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "France") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/France/%' and MrMrs like '/Miss/%' ") & " " & XmlAttribute("value", "2") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "Germany") & " " & XmlAttribute("filtertable", "amEmplDept") & " " & XmlAttribute("filter", "Location.FullName like '/Germany/%' and MrMrs like '/Miss/%' ") & " " & XmlAttribute("value", "4") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "<subgroup " & XmlAttribute("Name", "England") & " " & XmlAttribute ("filtertable", "amEmplDept") & " " & XmlAttribute ("filter", "Location.FullName like '/England/%' and MrMrs like '/Miss/%' ") & " " & XmlAttribute("value", "1") & " />" & Chr(13) & Chr(10) RetVal = RetVal & "</group>" </pre>

4. Validate the creation by clicking **Create** (Windows client) or **Save** (Web client).
5. If you are using the Windows client, you can preview the chart by:

- a. Click **Test**.
 - b. Press **ESC** to close the chart.
6. Close all windows after you finish.

Example 3 (advanced user)

In this, more complex example, data is exported from the Asset Manager database.

This displays a summary of all training sessions in the database sorted by training model:

- Number of training days
- Cost of training programs
- Number of training sessions: One training session per person participating in the session

These types of scripted statistics are always written according to same scheme. This example presents a commented model of a scripted statistic that you can test and modify as needed for your own statistics.

As previously, create a new scripted statistic (check the **Script** box).

Step 1: Define required variables

First, variables to be used to create the statistic must be defined.

The structure of the statistic must be designed:

1. The main table being used to extract and display the data is selected. In this example the selected table is the **Training** (amTraining) table.
2. Select one of the design options:
 - Grouping using multiple expressions.
 - Grouping using multiple criteria in a single expression.

In this example we will group data using **one criteria** (training model) with **three expressions** (the number of training days, the cost of the training sessions and the number of training sessions).

The columns correspond to the training models used for the main group.

The series correspond to the different expressions (the number of training sessions, the number of training days and the cost of the training sessions).

Enter the following text in the **Data script** field which is found on your statistic's **Script** tab:

```
SetMaxInst(1000000000)
const CONST_NumberSecondsPerMonth = 2592000
```



```
const CONST_NumberSecondsPerDay = 86400
Dim strRC as String
strRC = Chr(13) & Chr(10)
RetVal = ""
Dim strTable, strColumnField, strSerieField as String
strTable = "amTraining"
Dim strColumns, strSeries as String
strColumns = AmDbGetStringEx("SELECT DISTINCT Model.Name FROM amTraining WHERE
Model.Name <> ''", "x", "|")
strSeries = "Number of training sessions" & "|" & "Number of training days" & "|" &
"Cost"
```

Note: You can define:

- A **main filter** to select certain records from a given table.
- A **main expression** is used to perform operations on the records.

In this case, after retrieving the values for the different series (in step 2), you need to add the main filter to the filters used to retrieve the values, and add the main expression to each sub-group expression.

We will not use these functions in our example as they are not needed. Consult the database to see examples of these functions in use.

Note: When you are writing your statistic, if you want to perform operations on dates instead of on expressions as described in this example, then use the **StatLib** script library (this library can be accessed via the **Administration/ Scripts** link on the navigation bar).

You have three predefined functions that you can use to define the columns (strColumns):

CURRENTFISCALYEAR, LAST12MONTHS and **NEXT12MONTHS**.

To learn how these functions are used, read their descriptions in the script library (**Administration/ Scripts** link on the navigation bar).

Step 2: Retrieving values

In this step, values are retrieved for the different groups and series that have been defined.

- To retrieve values for the training model group in the **strColumns** variable, enter the following:

```
Dim iColumn, iColumnCount as Long
iColumnCount = CountValues(strColumns, "|")
For iColumn = 1 to iColumnCount
    Dim strColumn as String
    strColumn = GetListItem(strColumns, "|", iColumn)
    Dim lColumnValue as Long
```

- For each of the three series defined in **strSeries** (the number of training sessions, the number of training days and the cost of the training sessions), you need to define the filter (**strFilter**) and the expression (**strQuery**) to use to calculate the data and retrieve the values:

```
Dim iSerie, iSerieCount as Long
  iSerieCount = CountValues(strSeries , "|")
  dim strFilter as String
  dim strQuery as String
  dim lSerieValue as Long
  for iSerie = 1 to iSerieCount
    Dim strSerie as String
    strSerie = GetListItem(strSeries, "|", iSerie)
    if "Number of training sessions" = strSerie then
      strFilter = " Model.Name = " & AmSQLTextConst(strColumn)
      strQuery = "SELECT Count(lTrainingId) FROM amTraining WHERE " & strFilter
    elseif "Number of training days" = strSerie then
      strFilter = " Model.Name = " & AmSQLTextConst(strColumn)
      strQuery = "SELECT Sum(tsDuration)/" & CONST_NumberSecondsPerDay & " FROM
amTraining WHERE " & strFilter
    elseif "Cost" = strSerie then
      strFilter = " Model.Name = " & AmSQLTextConst(strColumn)
      strQuery = "SELECT Sum(mCost+mTax) FROM amTraining WHERE " & strFilter
    end if
    lSerieValue = AmDbGetLong(strQuery)
```

Step 3: Defining groups and sub-groups

Column data is defined by group and series by sub-groups.

```
dim subgroup as String
subgroup = subgroup & "<subgroup " & XmlAttribute("name", strSerie) & " " &
XmlAttribute("filtertable", strTable) & " " & XmlAttribute("filter", strFilter) & "
" & XmlAttribute("value", lSerieValue) & "/> "
lColumnValue = lColumnValue + lSerieValue
next
RetVal = RetVal & "<group " & XmlAttribute("name", strColumn) & " " & XmlAttribute
("value", lColumnValue) & "> "
RetVal = RetVal & subgroup
RetVal = RetVal & "</group> "
next
```

Step 4: Viewing the statistic

If you are using the Windows client, click **Test...** to display the statistic.

Defining custom colors for charted statistics

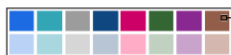
You can specify a set of custom colors to be used by a scripted query which returns charted statistics results for a set of subgroups or series. (It only makes sense to define colors in queries which involve multiple subgroups.)

To define a color for a series, use the **color=** parameter with the subgroup definition statement. For example

















```
<subgroup Name="Germany" filtertable="amEmplDept" filter="Location.FullName like '/Germany/%' and MrMrs='Mr'" value="9" color="#cc0066"/>
```

Note: To ensure compatibility with Asset Manager Windows and Web clients, colors need to be specified with the hex equivalent of their RGB values, for example **#6f006f** for purple.

Standard graphic colors are represented with the following values (however you can specify any other color with the hex equivalent of its RGB value) :



standard graphic colors

Color specifications for standard graphics		
 Blue R:28 G:107 B:227 (Hex:1C6BE3)	 Aqua R:51 G:166 B:182 (Hex:33A6E6)	 Gray R:156 G:156 B:156 (Hex:9C9C9C)
 R:186 G:210 B:246 (Hex:BAD2F6)	 R:168 G:215 B:223 (Hex:ABD7DF)	 R:215 G:215 B:215 (Hex:D7D7D7)
 Dark Blue R:15 G:71 B:128 (Hex:0F4780)	 Magenta R:204 G:0 B:102 (Hex:CC0066)	 Dark Green R:51 G:102 B:51 (Hex:336633)
 R:183 G:197 B:212 (Hex:B7C5D4)	 R:255 G:173 B:198 (Hex:FFADC6)	 R:193 G:209 B:193 (Hex:C1D1C1)
 Purple R:138 G:40 B:145 (Hex:8A2891)	 Brown R:46 G:93 B:73 (Hex:925D49)	
 R:202 G:163 B:205 (Hex:CAA3CD)	 R:212 G:183 B:174 (Hex:D4B7AE)	

This section includes:

Error handling	139
Practical Case	140

Error handling

The following rules are used to resolve anomalies in color definitions:

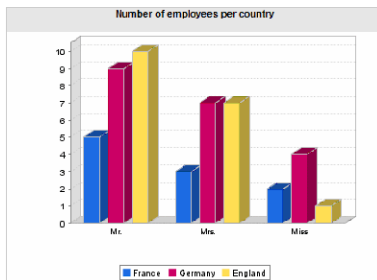
- If the **color=** parameter is undefined for one ore more subgroups in a query, default colors will be applied to those subgroups.
- If the **color=** parameter uses an unrecognized value, a default color will be applied.
- Colors can only be defined once per subgroup. (Any attempt to redefine them will be ignored.)

Practical Case

The following query defined below defines the customs colors blue, purple and yellow for three subgroups of the **Mr** group.

```
<group Name="Mr.">  
<subgroup Name="France" filtertable="amEmplDept" filter="Location.FullName like  
'/France/%' and MrMrs='Mr'" value="5" color="#1c6be3" />  
<subgroup Name="Germany" filtertable="amEmplDept" filter="Location.FullName like  
'/Germany/%' and MrMrs='Mr'" value="9" color="#cc0066"/>  
<subgroup Name="England" filtertable="amEmplDept" filter="Location.FullName like  
'/England/%' and MrMrs='Mr'" value="10" color="#ffde53"/>  
</group>
```

The query results in the following chart:



Chapter 7: Actions

This chapter explains how to define actions with Asset Manager.

Use the **Administration/ Actions** navigation menu item to create actions.

If you are using the Windows client, you can execute actions via the **Tools/ Actions** menu item or the contextual list of "Actions" in the toolbar.

If you are using the Web client, the actions are launched from the **Action** drop-down list on the table list screen.

This chapter includes:

Development best practices	141
Definition of an action	143
Creating an action	145
Examples of actions	153
Using variables	157
Actions linked to the helpdesk	157
Testing an action	159
Executing an action	160
Associating an action with a button in a screen	163
Overriding an action	163

Development best practices

This chapter includes:

Use a test functional domain to customize actions	142
Tag the Web services	143

Use a test functional domain to customize actions

Recommendation

We recommend that you associate a new action or a new version of an action (wizards, in particular) with a test functional domain before making it visible.

You can also mask the test functional domain to keep actions that are being developed hidden from users.

Justifications

As long as an action is incorrect, the functional domain to which it is linked is not generated as a Web service for Asset Manager Web.

Consequently, all objects (even valid ones) linked to the functional domain disappear from the Web services exposed by Asset Manager Web.

Implementation

To associate an action to a test functional domain:

1. Start the Windows or Web client.
2. Display the functional domains (**Administration/ Functional domains** link on the navigation bar).
3. Create the test functional domain.
To ensure that no Web service is generated for this functional domain, populate the **Web Service** (seWebService) field with the value **none**.
Note the SQL name of the test functional domain.
4. Display the actions (**Administration/ Actions** link on the navigation bar).
5. Link the action to the test functional domain (**Domain** link).
6. Once the object has been tested and validated, link it to the appropriate exposed functional domain.

Tag the Web services

If you modify script or wizard type actions, make sure you re-tag the Web services if you have deployed Asset Manager Web.

See **Tailoring** guide, chapter **Customizing the database**, section **Development best practices/ Tag the Web services**.

Definition of an action

An action enables you to automate, either completely or partially, the tasks performed on an Asset Manager database.

There are several types of actions:

- Executable
- DDE
- Messaging
- Script: modifying an object in the Asset Manager database
- Wizard
- Printing
- Software distribution
- Action
- Dashboard

Actions must first be defined in order for them to be executed by selecting them from a list.

Note: You can define a domain for an action, as well as categorize domains according to their functions using [functional domains](#).

Functional domain

Asset Manager enables you to define domains that group together the functions of the software. By default, certain functional domains are provided with the software: they correspond to the modules that you can activate or deactivate using the **File/ Manage modules** menu item in the Windows client.

Functional domains are used to create and categorize the information displayed in the Windows and Web client navigation bar. Thus, once you select a functional domain for an action, the action will appear in the Windows and Web client navigation bar under the heading of that functional domain.

Note: The contents of the navigation bar is reorganized and modified according to the context. If your action is contextual (it cannot be executed unless a specific screen is open, for example), then it will not appear in the navigation bar unless the current context corresponds to its definitions (if that specific screen is currently displayed, for example).

To define a functional domain:

1. Select the **Administration/ Functional domains** menu item.
2. Click **New**.
3. Assign a **Name** to your functional domain. This name is the one that will appear in the Windows and Web client navigation bar. By default, Asset Manager assigns an **SQL name** to the functional domain; you can modify this value if you want.
4. You can select a **Parent domain** for the functional domain as well if you want.
5. Validate your creation by clicking **Create** (Windows client) or **Save** (Web client).

Note: After you create a new functional domain, you may have to restart the Web Service to have it displayed on the web client.

Caution: If access to a functional domain is fully forbidden (read and write) for a user, they will not be able to access actions, views or reports in this domain. In practice, it recommended to authorize read access to the following tables when defining a functional domain:

- amFuncDomain
- amViewDef
- amReport
- amForm

- amAction
- amScriptLibrary

Creating an action

This section describes how to create an action:

Types of actions	145
General process of creating an action	149
Populating the DDE tab	150
Populating the Messaging tab	151

Types of actions

Asset Manager enables you to define several types of actions:

Executable actions

An **Executable** action causes a program to be executed.

It launches an **.exe**, **.com**, **.bat**, or **.pif** application. You can also refer to any type of document, as long as its extension is associated with an application in the file manager.

Note: Actions of the **Executable** type does not work on the web client.

DDE actions

DDE stands for "Dynamic Data Exchange"; it designates a method for dynamically exchanging information between programs. Asset Manager uses DDE have commands executed by another application.

A **DDE** action sends a DDE request to a DDE server application (or DDE compliant application) capable of handling DDE requests.

Example: Through DDE, you can request Microsoft Word to open a file whose name is specified and with given contents.

Messaging actions

Messaging actions allow you to send a message:

- Via Asset Manager's internal messaging system.
- Via an external VIM standard messaging system (Lotus Notes, Lotus cc:Mail, and so on).
- Via an external MAPI standard messaging system (Microsoft Exchange, Microsoft Outlook, and so on).
- Via an external SMTP standard messaging system.

Caution: You can only send messages via those messaging systems that you are able to connect to.

To issue a VIM, MAPI or SMTP standard message, Asset Manager uses:

- The **Account** (SQL name: MailLogin) and **Password** (SQL name: MailPassword) fields of the **Messaging** tab of the detail of the employee who logged in the Asset Manager database (table of Employees and departments) to identify the sender of the message,
- The **EMail** field (SQL name: EMail) in the **General** tab of the employee detail to identify the recipient of the message.

To send a message via Asset Manager's internal messaging system, Asset Manager uses the **Login** and **Password** fields in the **Profile** tab of the details of both the sender and recipient.

Note: The internal messaging address of an Asset Manager user is the same as the **Login**.

Caution: The administrator must create a user with the name "Admin" and populate the **Account**, **EMail** and **Password** fields to use an external messaging system and make sure that Asset Manager Automated Process Manager functions correctly.

Script actions

Script actions enable you to perform any operation on an Asset Manager database. They give advanced users extensive control over the database, allowing them to perform operations that cannot be performed with other types of actions, and in particular:

- Creating records
- Deleting records

- Duplicating records
- Modifying one or more objects in the Asset Manager database, such as all the records in a table, a field or a link.

The operations performed by this type of action are described by a Basic script; this enables you to use complex functions similar to those in the Asset Manager APIs.

Note: The complexity of the usable functions in **Script** action, associated with the ability to make in-depth changes to the database, make this type of action potentially dangerous to the database integrity. Therefore it should be used by advanced users only.

Different functions are used to change the value of a database object depending on the context of the action:

- If the action has no context, you must use functions derived from Asset Manager APIs such as **AmSetFieldStrValue()** or **AmSetFieldLongValue()**.
- If the action has a table as a context, you can use the **Set()** function; it has the following syntax:

```
Set [<Link.Link.Field>]=<Value>
```

Wizard actions

Wizards are complex actions. For further information, refer to the **Wizards** chapter of this guide.

Wizards are intended to guide you step by step through complex or recurrent tasks in Asset Manager. Wizards are designed via a dedicated programming language.

Printing-type action

A **Printing** type action enables you to print a report.

You need to populate the following fields for this type of action:

- **Type** field (seFormType): enables you to indicate the type of report to print.
- **Report** field: enables you to indicate which report must be used;

Note: The context of the action is determined by the context defined for the report.

Software distribution type action

A **Software distribution** type action enables you to create a scheduled task for a set of computers.

The context of these types of actions involves the **IT equipment** table (amComputer). These types of actions can only be executed if a computer is selected.

You cannot change the context.

The fields to populate for this type of action are found under the **Distribution** tab.

For more information about software distribution: see **Integration with software distribution and configuration management tools** guide.

To execute a **Software distribution** type action:

1. Display the computers (**Portfolio management/ Asset configurations/ IT equipment/ IT equipment** link on the navigation bar).
2. Select target computer groups (in this case, all computers in the group will be selected, even if they do not appear in the interface) and/or target computers of the scheduled task to be created.
3. Execute the action:
 - Windows client: **Tools/ Actions/ <Name of the action>** menu
 - Web client: select the <Name of the action> from the **Action** drop-down list on the IT equipment screen.

Only one scheduled task is created. The target of this task is each of the computers selected before executing the action whose **Software distribution identifier** (SWDID) is populated.

Action-type action

An **Action** type action enables you to execute an action on a set of records that were selected by a query.

The fields to populate for this type of action are found under the **Action** tab.

Example of application:

1. Create a **Software distribution** type action:
This action creates a scheduled task for a set of computers.
2. Create an **Action** type action to execute the **Software distribution** type action for the computers selected by the **Action** type action's selection query.
Use the selection query to have target computers selected automatically.

Dashboard-type action

A **Dashboard** type action allows you to create a dashboard containing statistics and reports. This action has no context.

The **Indicators** tab contains the list of the objects displayed by the dashboard. You can add or delete objects from this tab.

General process of creating an action

1. Select the **Administration/ Actions** navigation menu.
2. Click **New**.
3. Enter a name for the action.
4. In the **Type** field (SQL name: seActionType), specify the type of action you want to create. The type of action you select controls the display of one of the following tabs:
 - **Executable or DDE.**
 - **Messaging**
 - **Script**
 - **Wizard**
 - **Printing**
 - **Software distribution.**
 - **Action.**
 - **Indicators.**
5. You can populate the **SQL name** field (SQL name: SQLName) of the action detail, but it is not required. The SQL name is a unique way to identify the action; it is used in particular when executing an action via a DDE command (in cases where Asset Manager is used as a DDE command server).

Note: If you do not populate the **SQL name** field, it is automatically populated by the application.
6. Populate the **Context** field (SQL name: ContextTable):

- If you select a table from the drop-down list, the action is context-sensitive: It will only be proposed if you display the list of records in that table, or the details of one of those records.
 - If the action does not depend on a table, select the **(No table)** option at the top of the drop-down list.
7. Populate the **Domain** field, which enables you to specify the functional domain to which the action belongs. The action will appear under this domain in the Windows and Web client navigation bar.
 8. You can attach an icon to the action, but it is not required:
To do this, use the square located at the top left of the action detail screen. The image will then appear in the "Actions" context-sensitive list in the toolbar. The active icon in the list (the one displayed on the screen by default) is the icon for the last action executed using the toolbar.
 9. Populate the fields in the **Description** tab, and the fields in the specific tab for the "Type" of action you want to create.
 10. Click **Create** (Windows client) or **Save** (Web client).

Note: The Asset Manager administrator sees all actions, whether or not they are shared, and whether or not they were created by the administrator.

Populating the DDE tab

The information concerning a particular DDE action is located in the DDE tab of the action detail.

This tab is displayed only if you assigned the value DDE to the **Type** field (SQL name: seActionType) field in the basic information for the action.

The DDE mechanisms are based on the "services" provided by the software. In order to execute DDE mechanisms, you must define a "topic" that indicates the context in which the "commands" should be executed.

Therefore you must indicate:

- In the **Service** field (SQL name: DDEService), the name of the DDE service provided by the executable you want to call. This is usually a unique service for an executable. See the documentation of the executable for the list of services that it provides.
- In the **Topic** field (SQL name: DDETopic), the context in which the action should be executed.
- In the **Command** field (SQL name: DDECommand), the commands you want the external application to execute.
For Word, the command may be a Word Basic or a Visual Basic command.

If the DDE service of the application allows it, you can specify multiple commands. You must follow the syntax required by the external application.

- If the service is not present, indicate in the **File** field (SQL name: ActionFile) the file used to start the application that activates the service. This is the main application that responds to DDE commands.
- In the **Folder** field, specify the path of the file that activates the service.
- In the **Parameters** field, specify the parameters to be passed to the program running the DDE service.

Note: Commands sent to the external application are surrounded by square brackets ("[" , "]"). For example (using Microsoft Word):

```
[FileOpen("c:\tmp\test.txt")]
```

- When the action is contextual, you can use variables to reference the value of a field in the database. Since these variables are also surrounded by square brackets, Asset Manager cannot differentiate between commands and variables by itself. You must identify commands by prefixing the square brackets with a backslash character "\". Thus the previous example is written (in the case of a contextual action):

```
\[FileOpen("c:\tmp\test.txt")\]
```

You can combine commands and variables, as shown below (in this case the context is the table of assets):

```
\[FileOpen("c:\tmp\"+[AssetTag]+".txt")\]\[FileClose()\]\[FileExit()\]
```

- If the action is not contextual, the problem does not arise. Text surrounded by square brackets is still considered as commands to send to the external application.

Populating the Messaging tab

Information concerning a Messaging action is located in the Messaging tab of the action detail.

This tab is displayed only if you set to the **Type** field in the basic information of the action to **Messaging**.

Caution: In order for the system to function correctly, your system's PATH variable must include the folder containing the VIM DLL (VIM.DLL) and MAPI (MAPI.DLL).

What is the Referenced object field used for?

This field is used to select a link from the table selected in the **Context** field.

This field is only used for messages sent via Asset Manager's internal messaging system. It enables you to directly access the object that triggered the issuing of the message by simply clicking the **Referenced object** button in the message detail. When the referenced object is directly the record that triggers the action, you do not fill in the **Referenced object** field (SQL name: RefObject).

How to receive an acknowledgement

If you want the message sender to receive an acknowledgement via their usual messaging service, check the **Acknowledgment** box (SQL name: bAcknowledgment).

This acknowledgement will be sent to the address specified by the **Email** field (SQL name: EMail) in the **General** tab of the employee who opened the Asset Manager database (in the Employees and departments table).

Note: You cannot receive an acknowledgement for a message sent via the Asset Manager internal messaging system, or via a MAPI or SMTP messaging system.

How to send HTML format message

Note: The HTML format is only applicable to the SMTP protocol messages.

To send HTML format message, select the **HTML format** option (SQL name: bHTML) and enter the message contents with HTML tags.

How to indicate an address

Here are the different ways of indicating an address:

Address with form <Messaging engine>:<Messaging address>

<Messaging engine> can be:

- AM: to force the utilization of Asset Manager's internal messaging system.
- MAPI: to force the utilization of a MAPI standard messaging system (Internet Mail, Microsoft Outlook, and so on).
- VIM: to force the utilization of a VIM standard messaging system (Lotus Notes, and so on).

- SMTP: to force the utilization of a SMTP standard messaging system (Internet standard).

<Messaging address> has the usual form corresponding to the messaging system selected. Internal messaging addresses are the same as the "Logins".

Examples of addresses:

- AM:Admin
- MAPI:CathyBernard@taltek.com
- VIM:Cathy Bernard / TALTEK
- SMTP:cbernard@taltek.com

Address with form <Asset Manager login>

In this case, the messaging system used will be that which is indicated in the **EMail** (SQL name: EMail) of the **General** tab of the detail of the employee whose **Login (Profile** tab in the detail of the employee) is specified in the address.

If the **EMail** field is not populated, the message is sent via the internal messaging system.

For example:

1. A message is sent to the following Asset Manager logins: "Cathy", "Gerald" and "Philip".
2. The **EMail** fields show "MAPI:CathyBernard@taltek.com" for "Cathy" and "VIM:Gerald Colombo / Taltek" for "Gerald". The **EMail** field of "Philip" is empty.
3. If the sender has a MAPI account, the message will be sent to "Cathy" via MAPI and to the two other recipients via Asset Manager's internal messaging system.
4. If the sender has a VIM account, the message will be sent to "Gerald" via VIM and to the two other recipients via Asset Manager's internal messaging system.

Address with contextual variables

If the action is contextual, you can use variables between brackets []. These variables call on values of fields in the Asset Manager database.

For example: To send a message to the user of an asset selected in the table of assets, you can use **[User.Email]**.

Examples of actions

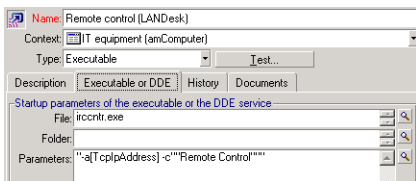
This section provides examples of Asset Manager actions:

Example of an executable-type action	154
Example of a DDE-type action	154
Example of Messaging type action	154
Example of a Script type action	155

Example of an executable-type action

The following screen describes a non-contextual action that launches Asset Manager Automated Process Manager and connects to a **amdemo** database:

Executable-type action - detail window



Example of a DDE-type action

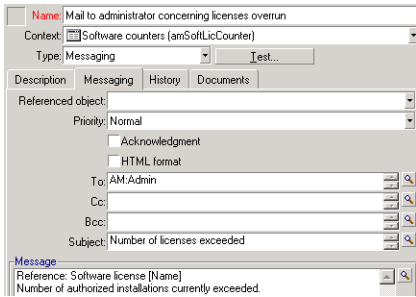
There are numerous applications of **DDE** type actions:

- Inserting Asset Manager data into a Microsoft Excel worksheet.
- Inserting information related to a purchase order in an accounting software package.
- Automatic sending of a confirmation message by fax on closing a ticket.
- Automatic sending of a work order request.

Example of Messaging type action

You send a message from the list of assets to indicate the expiration date of an asset's lease schedule to the user of that asset. This asset must have Lease listed as its mode of acquisition and must be linked to the lease schedule (**Acquis.** tab). In order for the referenced object to be the purchase request and not the request line, configure the action detail as follows:

Messaging-type action with a referenced object - detail window




Example of a Script type action

Creating a **Script** type action basically involves writing the Basic script that modifies the Asset Manager database.

Note: The use of functions specific to these actions is authorized within these scripts. These functions are indexed in the manual entitled "Programmer's Reference", chapter "Index of functions by field of application", section "Built-in functions"

Foreword

To prepare for creating the action, follow these steps:

1. Select the **Administration/ Actions** navigation menu and click **New** in the action screen.
2. Assign a name to the action you are going to create, such as "Test", and set the **Type** field (SQL name: seActionType) to **Script**. Do not select a context for the action. Click **Create** (Windows client) or **Save** (Web client).
3. If you are using the Windows client, you can click  in the **Script** tab to display the script builder window.

The programmable function, named `Success()`, used for these actions does not require an explicit return code. In the following example, we will create a new record in the Natures table based on the information contained in the table below:

Field label	SQL name of the field	Value of the field
Name	Name	Mini-computer

Field label	SQL name of the field	Value of the field
Created	seBasis	Portfolio item
Can be connected	bIsCnxClient	This box is checked

Writing the script

Enter the following example:

```
Dim lrec As Long
Dim lres As Long
lrec=AmCreateRecord("amNature")
lres=AmSetFieldStrValue(lrec, "Name", "Mini-computer")
lres=AmSetFieldStrValue(lrec, "seBasis", 1)
lres=AmSetFieldStrValue(lrec, "bIsCnxClient", 1)
AmInsertRecord(lrec)
```

Note: This action creates the desired nature without any user intervention.

Demonstration of the "Set()" function

Now we will create the same nature from a **Script** type action, by specifying the Natures table as the context for the action. In this case, we write the script as follows:

```
Set [Name]="Mini-computer"
Set [seBasis]=1
Set [bIsCnxClient]=1
```

Note: To execute this action, the user must open the Natures table and click **New**. After executing the script, the user must also click **Create** (Windows client) or **Save** (Web client) to validate the creation.

Tip


If you want to invalidate the execution of an action within a script, simply set the value of the return code to something other than 0 (12001, for example). This value is considered as an error code. The next command interrupts the action and cancels all changes already made:

```
RetVal=12001
```

Using variables

In the **Executable**, **DDE**, or **Messaging** tabs of the detail of a contextual action, you can use variables that reference the contents of fields, features or calculated fields in the database.

They use the form **[Link.Link.Field]**.

Tip: If you are using the Windows client, you can click the magnifier  to the right of the field to be populated for help in entering these variables.

Everything not contained within braces **[]** is considered as text.

For example: **[Link.Link.Field].doc**, calls the value of the field **Field** in the table linked to the main table going through the links **Link.Link**.

Caution: In order for the variables to work, the **Context** field of the action detail must show a table in Asset Manager and you must select a record in the list of records of the table before executing the action.

Actions linked to the helpdesk

Several main operations in the helpdesk can now automatically trigger actions (opening and closing a ticket, major changes, and so on).

For example, a message can be sent to the caller when a ticket is opened.

This section includes:

Actions defined by escalation schemes	157
Actions defined in the suspension slips	159

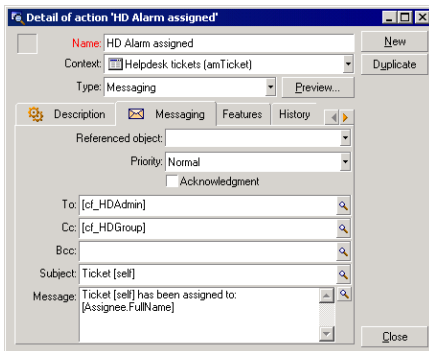
Actions defined by escalation schemes

The escalation scheme attached to the ticket defines the majority of the automatic actions associated with the ticket (**Escalation** (SQL field: EscalScheme) in the **Tracking** tab of ticket).

You must therefore define yourself the actions to be triggered according to the needs of your organization.

Here is an example of an action:

Messaging type action linked to an escalation scheme - detail window



As for all actions, you may insert variables concerning the contents of a given field. You may also use calculated fields designating a helpdesk team player:

Calculated fields designating a helpdesk team player

Variable	Description
cf_HDAdmin	Designates the helpdesk administrators.
cf_HDGroup	Designates the person responsible for the helpdesk group associated with the ticket.
cf_HDInCharg	Designates the ticket supervisor (assignee).
cf_HDContact	Designates the contact linked to the helpdesk ticket.
[self]	If the Context field (SQL name: ContextTable) is empty, returns the name of the action. If the Context field indicates the name of an Asset Manager table, returns the description string of the record selected in the table.

There is a hierarchy for these recipients: The Asset Manager administrator is hierarchically superior to helpdesk administrators, who are in turn higher up than ticket supervisors, in turn higher up than the contact for the ticket.

If one of these recipients does not exist for a ticket, then the message is sent to the recipient(s) immediately higher up in the hierarchy, and so on.

Note: The actions defined in the **Transitions** and **Assignment** tabs in the escalation scheme detail are triggered by an Asset Manager agent, whereas those defined in the **Alarms** tab are triggered by Asset Manager Automated Process Manager.

Actions defined in the suspension slips

When you suspend a ticket, you define:

- A maximum duration of suspension in the **Total suspension time** field (SQL name: tsTotalSuspTime).
- An action that Asset Manager Automated Process Manager triggers automatically if the ticket is still suspended when the suspension deadline is exceeded.

Testing an action

Note: Currently, this functionality is not available in the Web client.

To test an action when creating it, use the **Test** found in the top-right corner of the detail of the action to be tested.

Asset Manager displays a preview window in which you can select a context for the preview of the action. This window contains three buttons: **Calculate**, **Execute** and **Cancel**.

This section includes:

Calculate button	159
Execute button	159

Calculate button

Once the context is selected, click the **Calculate** button. This fills in the fields in the **Executable**, **DDE**, or **Messaging** tabs. Check that the variables have been correctly extracted from the record selected in the **Context** field (SQL name: ContextTable).

Execute button

This button enables you to execute the action directly from this screen.





Executing an action

This section includes:

Windows client	160
Web client	161
Multiple-selection in lists	161
Wizard-type actions	161
Executable type actions	162

Windows client

You may execute an action in one of several ways:

- Using the drop-down list  in the toolbar:
 - The button  is replaced by the icon associated with the last action used on this workstation, if this icon exists. If an action has already been executed, click the icon , or the icon replacing it to set it off again.
 - The  button displays the list of available actions.
 - To insert this pop-up list in the toolbar, use the Tools/ Customize toolbar menu item: it is part of the "Tools" category.
- Using the **Tools/ Actions** menu item: Click the desired action in the sub-menu.
- From the **Test** button found in the top-right corner of the detail of the action:
 - If the action is contextual, specify the Context (SQL name: ContextTable) by selecting a record in the action's reference table.
 - Click **Execute** to execute the action.
- From the shortcut menu (accessible by right-clicking). If at least one action is available for the open table, the **Actions** entry is shown in the shortcut menu.

Web client

You may execute an action:

- From the shortcut menu (accessible by right-clicking) on the list/ detail screen. If at least one action is available for the open table/ record, the **Actions** entry is shown in the shortcut menu.
- From the **Actions** drop-down list on the on the list/ detail screen.

Multiple-selection in lists

You may select several records in a list and apply an action to them.

In this way, you can select several assets and send the same message to their users.

Wizard-type actions

Wizards are composed of a succession of pages. Each of these pages displays information or requires user input, such as a selection to be made or data items to be entered.

Navigating between the different pages of a wizard is simple:

- Once the page is populated appropriately, you can move to the following page (determined by a transition) by clicking the **Next** button. This button is not available for the final page.
- You can always go back to make any corrections by clicking the **Previous** button.
- You can execute the final action of the wizard at any given moment by clicking the **Finish** button. If the wizard does not have sufficient information to perform its designated task, the appropriate page is displayed.

You can cancel the execution of a wizard completely (and as a consequence, its associated action) by clicking the button **Cancel**.

Executable type actions

The behavior of Executable type actions depends on the value of the **File** (ActionFile) field of the action's detail.

The following tables summarize the possible behaviors for the Windows and Web clients depending on the situation:

If the **File** (ActionFile) field of the action's detail starts with **http** or **https**

If the action is triggered by a Windows client	If the action is triggered by a Web client
Starts the default Internet browser on the client's Windows workstation and connects to the URL address provided in the File field (ActionFile)	Displays the page corresponding to the URL address specified in the File (ActionFile) field of the Web client's workspace.

If the **File** (ActionFile) field of the action's detail starts with **ftp**

If the action is triggered by a Windows client	If the action is triggered by a Web client
Starts the Windows explorer on the client's Windows workstation and connects to the ftp site provided in the File (ActionFile) field.	Displays the ftp site corresponding to the URL address specified in the File (ActionFile) field of the Web client's workspace.

If the **File** field of the action's detail starts with **mailto**

Starts the default messaging application on the Windows or Web client's workstation and creates a new message using the parameters specified after **mailto:** in the **File** field (ActionFile).

If the **File** field of the action's detail starts with another value

If the action is triggered by a Windows client	If the action is triggered by a Web client
Executing the file specified in the File (ActionFile) field	By default, the file is not executed on the Asset Manager Web Service station nor on the Web client and an error is returned. To ensure the file is executed on the Asset Manager Web Service station:

If the action is triggered by a Windows client	If the action is triggered by a Web client
	<ol style="list-style-type: none"><li data-bbox="605 338 1308 401">1. Display the database options (Administration/ Database options... menu on the Windows client).<li data-bbox="605 422 1308 485">2. Select the Actions/ Executing executable-type actions (ExecuteAction) option.<li data-bbox="605 506 932 527">3. Set this option to Server. <p data-bbox="626 575 1211 606">Caution: This type of operation carries some risks.</p>

Associating an action with a button in a screen

There are several ways to associate an action with a button in a screen.

For more information, see:

- The **Tailoring** guide, part 1 **Customizing client computers**, chapter **Customizing a client workstation**, section **Customizing buttons**
- The **Tailoring** guide, chapter **Customizing the database**, section **Customizing existing objects/ Customizing objects / Customizing a detail/ Creating action buttons**.
- The **Tailoring** guide, chapter **Customizing the database**, section **Creating new objects/ Creating action buttons**.

Overriding an action

The "Overriding" feature allows you to override and customize system actions (scripts or wizards) delivered with Asset Manager.

For example, if you want to modify the functionality of the **Ticket Closure** action, you can simply duplicate the existing system action **Ticket Closure** into a new action **My Ticket Closure**, modify the duplicated action, and then configure the default **Ticket Closure** action to be overridden by **My Ticket Closure**. After you do this, every time you launch the **Ticket Closure** action, the customized action **My Ticket Closure** will be run instead of the default action.

This feature is completely transparent to the users. If you override a system action with a customized action, the action name on the client menu will not be changed, though the actual executed action is the customized one.

Note: The "Overriding" feature is only available on the Windows client.

To use this feature, follow these steps:

1. On the **Tools** menu, point to **Actions**, and then click **Edit**, the **Detail of Action** window pops up.
2. Select an action, the **Overridden By** combo box is available in the right pane.
3. Open the **Overridden By** combo box, you will see all customized actions that can be used to override the selected system action. Select the expected action, and save the change.

Note: The customized action must have no context table or have the same context table as the system action. Otherwise, an error will occur. In addition, after you override an action, the customized actions domain cannot be changed.

Known issue

When you try to open a legacy database by using an Asset Manager 9.61 client, the client generates the following error:

"Field 'IOverriddenById' is unknown in table 'Actions (amAction)'."

If you then open the **amAction** screen and launch an action, the client crashes.

This issue occurs because the Overriding feature tries to access a field that does not exist in older versions of Asset Manager.

Chapter 8: Workflow

Note: This chapter fully applies to the Windows client. However, some of the functionalities are currently not available in the Web client.

The aim of workflow is to organize business processes and how they are implemented. To use the definition of the **WfMC** (Workflow Management Coalition - an non-profit organization whose mission is to define, develop and promote standards for workflow solutions), a process is "A set of one or more linked procedures or activities which collectively realize a business objective or policy goal". Managing these processes implies modeling working procedures and taking into account all items linked to the working of the organization.

The workflow system in Asset Manager:

- Makes it possible to formalize business processes using **workflow schemes**: Who does what, when and how
- Supports the execution of **instances** of these processes in compliance with the formal definition
- Makes available the information and tools required to execute **tasks**, which are defined in activities and associated with processes. A task may be executed automatically or require the attention of a user.
- Offers metrics on the instances carried out and in progress

For example, the following processes can be modeled and automated using workflow methods:

- Purchase-request approval procedures.
- Asset moves.

Creating a workflow scheme in Asset Manager consists of defining:

- A start activity, which constitutes the starting point of the process,
- Events (contained in an activity),
- Transitions generated by events,
- Activities triggered by transitions,
- A context,
- Time limits and alarms.

Workflow - simplified scheme



This chapter includes:

Definitions	166
General overview	167
How to implement workflow	168
Using the graphical workflow editor	169
Example of workflow used in request approval	171
Context of a workflow	185
Workflow roles	187
Workflow activities	188
Workflow tasks	193
Workflow events	197
Workflow transitions	208
Workflow alarms and time limits	208
Workflow execution groups	210
Workflow tracking	211
Deleting instances of finished workflows	213
Technical information: Data model	218

Definitions

This section defines several key terms used in workflow:

Workflow instance

A workflow instance refers to a workflow that is being executed according to the formal process defined in a workflow scheme.

Workflow activity

A workflow consists of:

- A task to be executed. This task may necessitate user interaction or be carried out automatically by Asset Manager Automated Process Manager.
- Events that trigger transitions to other activities.

Workflow event

Workflow events results from activities. They in turn make it possible to activate transitions that trigger other activities.

Workflow transition

A workflow transition makes it possible to go from one activity to another. They are triggered by an event.

An event can be associated with several transitions.

Workflow task

A workflow task is an assigned task to be carried out, resulting from the triggering of an activity.

Workflow activity assignee

Activity assignees the users responsible for performing tasks resulting from workflow activities.

Workflow execution group

Workflow execution groups enable you to categorize the workflow schemes that you define. The execution group to which a given workflow scheme belongs is indicated in the detail of the workflow scheme.

General overview

A key step in implementing workflow under Asset Manager is to define workflow schemes using the graphical editor via the **Administration/ Workflows/ Workflow schemes** navigation menu item.

Workflow schemes define activities, events and transitions. They can reference Asset Manager actions and employees (workflow assignees).

Workflow schemes are interpreted by workflow engines. The workflow engines in question are run by either Asset Manager Automated Process Manager, or agents of Asset Manager.

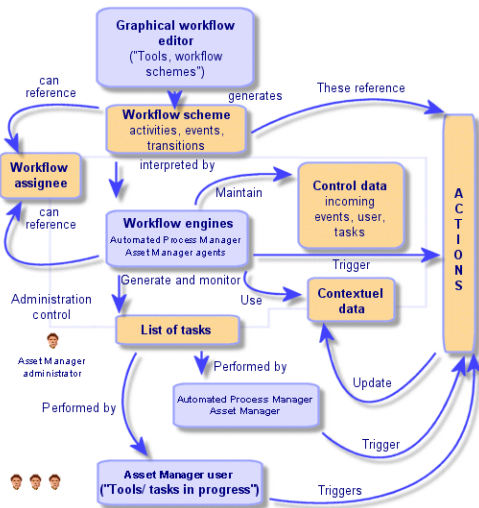
In reaction to events, workflow engines trigger and monitor workflow instances:

- The workflow engines generate the tasks to be performed.
- They monitor these tasks and events leading to activities.
- They also keep a record of the course of events, by logging incoming events and user tasks to be performed.

Workflow tasks are either performed by the workflow engines or by Asset Manager users. Once they are carried out, they activate events that are then taken into account by the workflow engines.

The following diagram gives an overview of how workflow is implemented in Asset Manager:

Workflow in Asset Manager - overview



The processing of workflow instances varies according to the nature of activities and events and the way that the workflow engines controlling them have been configured.

How to implement workflow

Workflow under Asset Manager makes it possible to implement company procedures. Here are the steps to follow:

1. Analyze the procedures of your company that you want to formalize.
2. Create:
 - Workflow roles.
 - Actions.
3. Create workflow schemes for which you define:
 - Activities, events and transitions.
 - Alarms.
4. Define the appropriate workflow execution groups. Associate each workflow scheme with a workflow execution group.
5. Launch Asset Manager Automated Process Manager on one or more machines. For each instance of Asset Manager Automated Process Manager, define the workflow execution groups to be monitored and the monitoring parameters.

Using the graphical workflow editor

Use the **Administration/ Workflows/ Workflow schemes** navigation menu item to access the workflow schemes. This menu item is reserved for Asset Manager administrators.

The **Activities** tab in the detail of workflow scheme includes a graphical editor that enables user-friendly creation of the workflow schemes.

This section details how you use this graphical editor to create, modify or delete workflow schemes:


Activities	169
Events	170
Transitions	171
Other functionality	171

Activities

To create an activity:

- Click the  button, then click in the graphical zone. The detail of the activity is displayed.

To delete an activity:

- Either select the activity by clicking it then press the "Delete" key.
- Or select the activity and click the  button.

To modify the detail of an activity:

- Double-click the activity.




Events


The events contained in the activities are of two types.

Database or Periodical type event

To add an output event to an activity:

- Select the activity, then click the  button.

To delete a **Database** or **Periodical** type output event from an activity:

- Either select the event, then press the "Delete" key.
- Or select the event and click the  button.

To modify the detail of a **Database** or **Periodical** type event:

- Double-click the event.

System event

There are two different types of events in the application; System events and user events. System events are events triggered by the application as the result of an activity. User events are triggered by the user, for example, as the result of a question-type event.


The event type is defined in the **seType** field (field not visible).

Transitions

To create a transition:

1. Click the starting event to select it.
2. Hold the mouse button down and drag to the target activity.

To delete a transition:

- Either click the transition to select it, then press the "Delete" key.
- Or select the transition and click the  button.

To modify the detail of a transition:


- Double-click the transition to display the detail of the transition.

To modify the source and/ or target of a transition:

1. Select the transition.
2. Drag the end you want to modify.

Other functionality

The graphical editor also allows you to:

- Drag and drop linked activities and transitions.
- Enlarge or reduce the scheme using the **Zoom** slider or the  button.

Example of workflow used in request approval

This section details a simple example of workflow:

Aim	172
Prerequisites	173
Creating activities	176

Configuring events created at the same time as activities	180
Creating the start event	181
Creating transitions	181
Example of activating a workflow instance	182

Aim

The aim of this workflow scheme is to automate the purchase request process according to the following:

Workflow - request validation



The steps of the workflow scheme are as follows:

1. The workflow instance starts as soon a purchase request is to be validated (1), that is, when the **Req. status** field (SQL name: seStatus) in the purchase request detail indicates **Awaiting approval**.
2. The request first undergoes Technical validation (2). This step consists of submitting the request to the departmental supervisor for approval. They are informed by a message. A reminder alarm is programmed to be triggered if the request is not dealt with by the approver before the end of the next business day following the issuing of the approval request message.
3. If the person responsible validates the purchase request (3), the next step is financial validation. Otherwise the request is refused (3b).
4. Financial validation (4) consists of submitting the request to the company's financial controller, Mr. Gerald Colombo. He is also notified by mail and a approval reminder alarm is programmed.
5. If the financial controller validates the purchase request, the purchase request is approved (6). Otherwise the request is refused (5b).
6. If the purchase request is approved, Asset Manager sets the **Req. status** field in the purchase request detail to **Validated**.
When the purchase request is refused (6b), Asset Manager sets the **Req. status** field in the purchase request detail to **Refused**.

Prerequisites

You must connect to the database under the login **Admin** and configure the messaging system (see **Administration** guide, **Messaging** section).

Creating workflow assignees

The assignees involved in this workflow scheme are:

- The departmental supervisor of the requester.
- The financial controller of the company, Mr. Gerald Colombo.

Workflow assignees are employees defined in the Employees and departments table and who are authorized to interact at one or more stages of a workflow.

The workflow assignees are created and stored in the Workflow Roles table (SQL name: amWfOrgRole). To create them, select the **Administration/ Workflows/ Workflow roles** navigation menu item, then click **New**.

Departmental supervisor of requester

This person is calculated by a script. To define this, populate the detail screen as follows:

- Designation: Departmental supervisor of the requester
- Context: Requests (amRequest)
- Type: Calculated individual
- Script: RetVal = [Requester.Parent.Supervisor]

Financial controller

This person is designated as Mr. Gerald Colombo. To define him as an assignee, populate the detail screen as follows:

- Designation: Financial controller
- Context: (No table)
- Type: Designated individual
- Assignee: Colombo, Gerald

Creating actions

The workflow scheme calls on numerous actions. To create them, select the **Administration/ Actions** menu item.

Request for technical validation sent to departmental supervisor of requester

This action is used in the technical validation phase. It makes it possible to notify the person in charge of technical validation of the need to review this request. To create this action, populate the detail screen as follows:

- Name: Request for technical validation
- Context: Workflow tasks (amWfWorkItem)
- Type: Messaging
- Messaging tab:
 - Priority: Normal
 - To: [Assignee.Email]
 - Subject: Request approval
 - Message: You need to refuse or approve a purchase request.

Request for financial validation sent to financial controller

This action is used in the financial approval process. It sends a message to the person in charge of financial validation of the need to review this request. To create this action, populate the detail screen as follows:

- Name: Send a financial-validation reminder
- Context: Workflow tasks (amWfWorkItem)
- Type: Messaging
- Messaging tab:
 - Priority: Normal
 - To: [Assignee.Email]
 - Subject: Request approval
 - Message: Do you approve the request [ReqNumber]?

Validation of the purchase request

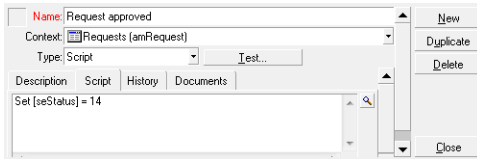
This action is used at the level of the **Request approved** activity, which will be created later on.

It sets the **Req. status** field (SQL name: seStatus) in the purchase request detail to **Validated**. This action is a **Script** type action.

The **Req. status** field is a system itemized list. To see the list of its values, display the help on this field:

1. Move the cursor on the **Req. status** field in the detail of the request.
2. Focus on this field and press **Shit+F1**: The value displayed as **Validated** is stored in the database as **14**.

If you are using the Windows client, the action is as follows:

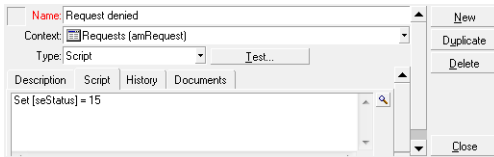


Refusal of the purchase request

This action is used at the level of the **Request denied** activity, which will be created later on.

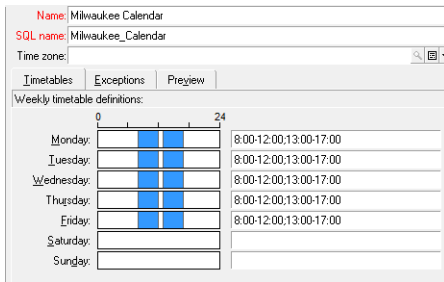
It is similar to the **Request approved** action, but the **Req. status** field (SQL name: seStatus) in the **General** tab of the request detail must be set to **Refused**.

If you are using the Windows client, the action is as follows:



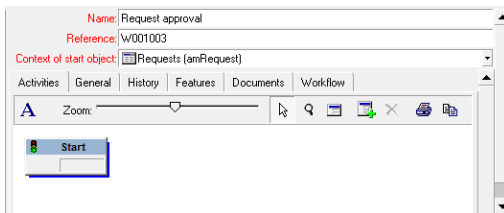
Creating the calendar

Use the **Tools/Calendars** menu item to access the list of calendars. This calendar is associated with workflow scheme activities. It enables you to set deadlines for instances of the workflow scheme:



Preparing the workflow scheme

1. Launch the **Administration/ Workflows/ Workflow schemes** menu item.
2. Click **New**.
3. Assign the name **Request approval** to the workflow.
4. Indicate the context of the start object that will apply by default to all activities in the workflow scheme. In our case, this is the table of requests (SQL name: amRequest).
5. Click **Create**: The start activity (**Start**) is automatically created by Asset Manager in the graphical editor in the **Activities** tab.



Creating activities

Activities can be created graphically in the **Activities** tab of the workflow detail:

1. Click .

Creating the **Technical validation** activity

1. Assign a name to the **Technical validation** activity.

2. Since the activity submitting the request to the departmental supervisor for approval, select the value **Question** from the **Type** itemized list (SQL name: seType).
3. The **Context** field (SQL name: ContextTable) in the **General** tab is not modified.

Configuring the Technical validation activity

1. Populate the **Parameters** tab as shown below:

The screenshot shows a configuration window for a 'Technical validation' activity. The 'Name' field is 'Technical validation' and the 'Type' is 'Question'. The 'Reference' is 'A001010'. The 'Person or group in charge of the activity' is 'Departmental supervisor of the requester'. The 'Action' is 'Request for technical validation'. The 'Question or instructions' field contains 'Do you approve or refuse the request [ReqNumber]?'. The 'Possible answers' section shows 'Designation: Approve' and 'SQL name: Approve'.

2. Specify the question to be asked:
 - a. The text of the question references the number of the purchase request.
 - b. There are two possible responses: Refuse or approve. To add a sub-tab describing a response to the question, right-click the label zone of the sub-tabs and select **Add linked record**.
3. Indicate to whom the question is addressed in the **Person or group** field (SQL name: Assignee). In our case, the assignee is the departmental supervisor of the requester. This person was created in the table of workflow roles in the preliminary steps.
4. In order for the assignee to be notified of the need to review this request:
 - a. Set the **Notify person or group** field (SQL name: bNotifAssignee) to **Yes**.
 - b. Specify the action to be carried out: This is the **Request for technical validation** action created in the intermediary step. This action is automatically triggered when a purchase request is submitted to technical approval.

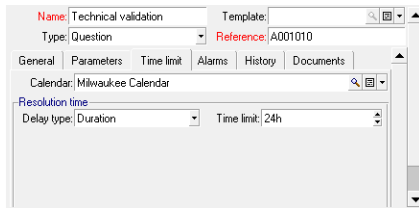
Defining the time limit for performing the Technical validation activity

Note: You have to select the **Log task** option in the **General** tab before you can define the time limit.

In the **Time limit** tab of the activity detail:

1. Specify the calendar of business days attached to the activity. This calendar is taken into account when calculating time limits. Select the **Milwaukee calendar** created in the preliminary step.

2. Define the time after which the decision must be taken in relation to the time at which the activity is triggered. In our case, the assignee must respond to the question within 24 hours.



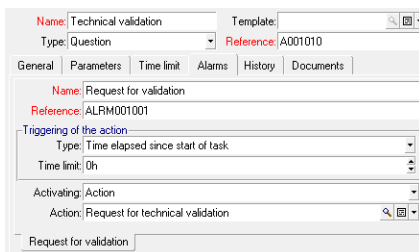
Defining an alarm for the Technical validation activity

Note: You have to select the **Log task** option in the **General** tab before you can define an alarm.

In the **Alarms** tab of the activity detail, define a reminder alarm in case the decision is not taken before the end of the time limit specified in the **Time limit** tab.

In order to simplify things, the alarm will trigger the **Request for technical validation** action:

It is possible to define further alarms using the **Add linked record** command from the shortcut menu.



Events

Once the activity is created, Asset Manager creates two system events **Approve** and **Refuse**

corresponding to the two possible responses to the question: 

When these events occur, an Asset Manager agent logs them in the table of workflow elementary events (SQL name amWfOccurEvent).

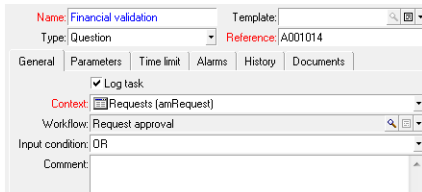
The following activities are triggered either by Asset Manager directly or by Asset Manager Automated Process Manager according to the configuration of the workflow events:

- If the **Processing** field (SQL name: seProcessingMode) is set to **Log event and process immediately** or **Process event immediately without logging**, Asset Manager triggers the following action.
- If the **Processing** field (SQL name: seProcessingMode) is set to **Log event and process by server**, Asset Manager Automated Process Manager triggers the following action.

By default, the (**Processing** field (SQL name: seProcessingMode) in the **General** tab of event detail is set to **Log event and process by server**.

Creating the **Financial validation** activity

This activity is similar to the previous one.



The screenshot shows the configuration form for the 'Financial validation' activity. The 'Name' field is 'Financial validation' and the 'Reference' is 'A001014'. The 'Type' is 'Question'. The 'Context' is 'Requests (anRequest)'. The 'Workflow' is 'Request approval'. The 'Input condition' is 'OR'. The 'Log task' checkbox is checked. The 'Comment' field is empty.

- The assignee differs: In this case, it is the financial controller of the company, Mr. Gerald Colombo (a designated individual). He was created in the table of workflow roles in the previous step with the role of Financial Controller. In order to notify him, select the **Send a financial-validation reminder** action created previously (**Action** field).
- Delays and alarms are created in much the same way as for the Technical validation activity.

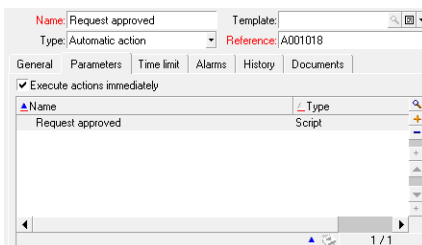
Creating the **Request approved** activity

When the request succeeds in passing the two validation steps, it is approved.

The **Request approved** activity is one of the possible endings of the workflow scheme.

This activity needs to modify the detail of the request to show the request as being approved.

This activity is thus an **Automatic action** type activity: The action to be executed is the **Request approved** action created in the preliminary step.

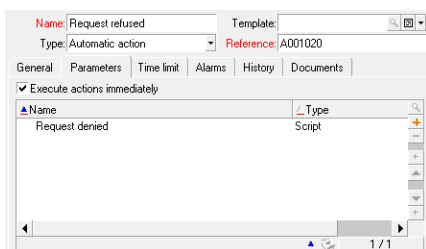


The screenshot shows the configuration form for the 'Request approved' activity. The 'Name' field is 'Request approved' and the 'Reference' is 'A001018'. The 'Type' is 'Automatic action'. The 'Execute actions immediately' checkbox is checked. The 'Action' list contains one entry: 'Request approved' with the type 'Script'.

Creating the **Request refused** activity

The **Request refused** activity is similar to the **Request approved** activity.

In this case, the detail of the request needs to be modified to indicate that the request has been refused. The action to be executed is the **Request denied** action created in the preliminary step.



Configuring events created at the same time as activities

When the activities were created during the previous steps, the following events were also created:

- **Technical validation** activity:
 - **Approve** event
 - **Refuse** event
- **Financial validation** activity:
 - **Approve** event
 - **Refuse** event
- **Request approved** activity
 - **Executed** event
- **Request refused** activity:
 - **Executed** event

Note: You can only perform the following tasks in the Windows client.

If you select each event one after the other and perform the following options, then you won't have to rely on the Asset Manager Automated Process Manager:

1. Double-click on the event name.
2. Select the **General** tab in the event detail screen.
3. Select the **Log event and process immediately** value to save the **Processing** field (SQL name:

seProcessingMode).

4. Click **Modify**.

Creating the start event

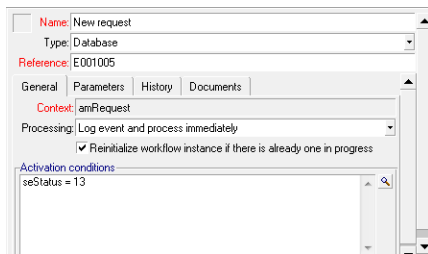
Events that trigger a workflow instance are associated with the **Start** activity.

To define a **Start** event, double-click the empty event zone in the **Start** activity.

Note: You can only perform the following tasks in the Windows client.

1. In our case, the workflow instance is triggered when the **Req. status** field (SQL name: seStatus) in the detail of a request detail is set to **Awaiting approval**.

The start event is therefore a **Database** type event and its parameters are described in the **General** tab as shown in the screen below:



2. Set the **Processing** field (SQL name: seProcessingMode) in the **General** tab of the event to **Log event and process immediately**.
3. In the **Parameters** tab of the event:
 - a. Check the **Post-Update** box (SQL name: bUpdate).
 - b. Specify the **seStatus** field in the **Fields monitored** field (SQL name: MonitFields).

Creating transitions

Once the activities are created, they now need to be linked by transitions.

Note: You can only perform the following tasks in the Windows client.

To create a transition:

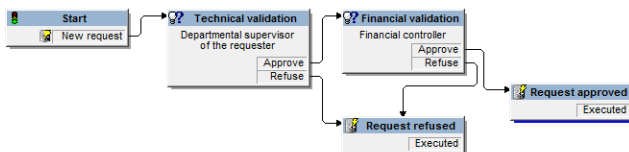
1. Click the start event of the transition.
2. Hold the button down and drag to the target activity.

In this case, the transitions to be created are the following:

- From the start event to the **Technical validation** activity.
- From the **To approve** event of the **Technical validation** activity to the **Financial approval** activity.
- From the **To approve** event of the **Financial validation** activity to the **Request approved** activity.
- From the **Refuse** events of the **Technical validation** and **Financial validation** activities to the **Request refused** activity.

We obtain the following workflow scheme:

Workflow - request validation scheme



Example of activating a workflow instance

We now need to verify that the workflow scheme functions correctly.

In order to do this, we need to do the following:

Populating the Employees and departments table

Before creating the purchase request to be approved, it is important to define the requester and the corresponding departmental supervisor in the table of departments and employees. The supervisor needs to have the appropriate rights to perform the following operations:

1. Create the requester **Jerome Carpenter**, belonging to the **IS department**.
2. Assign the user name (login) **Carpenter**, a password and a user profile allowing him to enter a purchase request (**Profile** tab in the detail of the corresponding record). You can use the **Requester** profile in the demonstration database.

3. Assign the value **AM:Carpenter** to the **Email** field (SQL name: EMail).
4. The supervisor of the **IS department** is **Philip Chavez**.
5. To simplify the following operations, grant administrative rights for the database to Philip Chavez: Display the **Profile** tab in the detail of the corresponding record and check the **Administration rights** box (SQL name: bAdminRight). Specify the **Login** (SQL name: UserLogin) and the password of Philip Chavez.
6. Assign the value **AM:Chavez** to the **Email** field (SQL name: EMail).
7. Select the employee **Gerald Colombo**.
8. Assign the value **AM:Colombo** to the **Email** field (SQL name: EMail) and **Colombo** to the **Login** field (SQL name: UserLogin).

Creating a purchase request to be approved

The following step consists of creating a purchase request to be approved:

1. Connect to the demonstration database using the login name of Jerome Carpenter.
2. Launch the **Asset lifecycle/ Procurement lifecycle/ Requests/ Purchase requests** navigation menu item under Asset Manager.
3. Click **New**.
4. In the **Requester** field (SQL name: Requester) in the **General** tab of the detail of the request, select the record **Carpenter**.
5. Set the **Req. status** field (SQL name: serStatus) in the detail of the request to **Awaiting approval**.
6. Confirm the purchase request: The start event occurs and an Asset Manager logs the event in the table of workflow elementary events (SQL name: amWfOccurEvent).

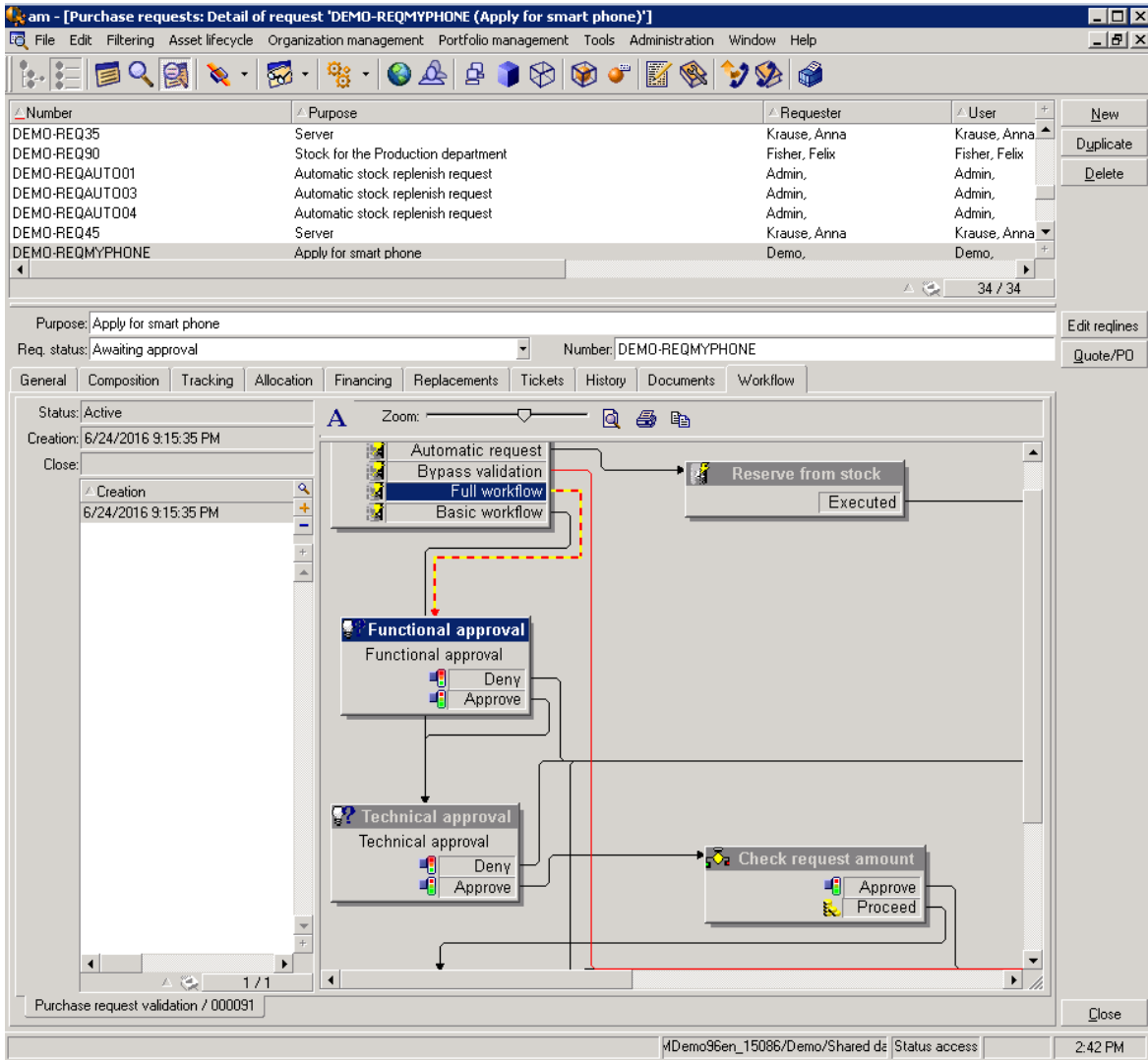
Controlling the instance

In order to verify that the workflow instance is functioning correctly, launch Asset Manager and connect to the demonstration database using the login of Philip Chavez, the departmental supervisor of "Jerome Carpenter".

Viewing the workflow instance

Display the detail of the purchase request that you have created: The **Workflow** tab lists the current workflow instances for the request. Each instance is described in an individual sub-tab.

- The left part of the sub-tab list the events that have occurred.
- The right part shows the status of the instance. In our case, the **Technical validation** task should be flashing.



Viewing the task to be carried out

1. Select the **Asset lifecycle/ Infrastructure management/ My workflow tasks** navigation menu item: The departmental supervisor "Philip Chavez" can thus view the detail of the assigned task.
2. The **General** tab displays the question that you have defined in the **Question** tab of the **Technical validation** activity.
 The **Assignment** tab describes who is in charge of the task and the corresponding deadline. This date is calculated using the information contained in the **Time limit** tab in the detail of the **Technical validation** activity, and the date of creation of the task (that is, the date of activation of

the transition by Asset Manager Automated Process Manager).

You can click **Detail** to access the detail of the request giving rise to the task.

3. Simply click either the **Approve** or **Refuse** buttons to carry out the task. It is also possible to enter a comment concerning the decision in the **General** tab.

Context of a workflow

Each workflow instance has its own specific context

This section includes:

Defining the context of a workflow instance	185
Object referenced by a workflow instance	185
Limiting workflow instances in progress for a given object	186

Defining the context of a workflow instance

When defining a workflow scheme, you define:

- A default context for the workflow.
- A context for all events, transitions and workflow activities (in the detail screen of a transition or in the **General** tab of the detail of events and activities). This context is linked to the default context of the workflow instance.

In both cases, the context is a table.

Object referenced by a workflow instance

When a record fulfills the activation conditions defined in a workflow scheme, a workflow instance is triggered. The record constitutes the object referenced at start event level.

When the workflow instance is running, the referenced object can change according to the context defined at activity, event and workflow transition level.

Example: A workflow instance is triggered when a purchase request is approved. It creates a purchase order according to this request. If request R1 is approved, it constitutes the referenced object of the

start event. The referenced object then becomes purchase order PO1, that is, the order generated from the purchase request.

Limiting workflow instances in progress for a given object

One single active workflow instance for an object option (SQL name: bUniqueActive)

Asset Manager makes it possible for you to limit the number of concurrent workflow instances for a given object using the **One single active workflow instance for an object** option in the **General** tab of a workflow detail.

If an output event of the "Start" activity giving rise to a second workflow instance for an object occurs, the **One single active workflow instance for an object** and **Reinitialize workflow instance if there is already one in progress** options (SQL name: bReinitialize) (**General** tab of the detail of the event), determine the outcome:

The following table resumes the different possible cases:

		One single active workflow instance for an object option in the General tab of the workflow scheme.	
		Validated	Not validated
Reinitialize workflow instance if there is already one in progress option in the General tab of the output event of the Start activity.	Validated	If there is already a workflow instance in progress for the object, it is stopped and a new workflow instance started.	
	Not validated	If there is already a workflow instance in progress for the object, the event is ignored (no new workflow instance).	A new workflow instance is created.

Example of application

In the case of a workflow scheme intended to deal with purchase request approvals, it may be useful to:

- Check the **One single active workflow instance for an object** option, in order for a given purchase request to be subject to one single approval process.
- Check the **Reinitialize workflow instance if there is already one in progress** option at the level of the start event to restart the instance if the composition of the purchase request is modified.

Workflow roles

Tasks resulting from certain workflow activities must be carried out by an assignee.

Note: Activity assignees are only involved in **Question** or **User action** type activities. **Automatic action** or **Test / script** type activities do not have assignees.

Activity assignees are selected in the workflow roles table (SQL name: amWfOrgRole). Use the **Administration/ Workflows/ Workflow roles** link on the navigation bar to access the workflow roles table.

Workflow role type

There are several possible types of workflow roles (**Type** field (SQL name: seType) in the workflow role detail):

Designated individual

In this case, the assignee is selected in the table of departments and employees directly.

Example:



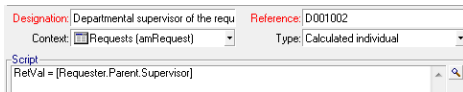
The screenshot shows a form with the following fields:

Designation: Financial controller	Reference: D001004
Context: (No table)	Type: Designated individual
Assignee: Colombo, Gerald	

Calculated individual

In this case, the assignee belongs to the table of departments and employees but is calculated by a script.

Example:



Group

In this case, the **Person or group** field (SQL name: Assignee) is selected in the Employee Groups table (amEmplGroup).

Calculated group

In this case, the **Person or group** field (SQL name: Assignee) is calculated by a script in the Employee Groups table (SQL name: amEmplGroup).

Defining the assignee of an activity

The **Person or group** field (SQL name: Assignee) in the **Parameters** tab lets you define the assignee for **Question** and **User action** type of activity.

Workflow activities

Activities can be divided into two groups:

- Those requiring the interaction of a user: **Question** and **User action** type activities (**Type** field (SQL name: seType) at the top of an activity detail).
- Those that are performed automatically: **Automatic action** and **Test / script** type activities.

The value of the **Type** field of an activity determines which tabs are displayed in the activity detail.

This section describes the following activities:

User-action type activity	189
Question type activity	189
Automatic-action type activity	190
Test/ Script type activities	191
Start activity	192
Activity templates	192

User-action type activity

These activities require the involvement of a user, named the "assignee". The assignee is shown in the **Person or group** field (SQL name: Assignee).

Their definitions include:

- Instructions to follow.
- A wizard to execute.

Specify:

- The instructions to be followed.
- The wizard to be executed.
- The record in the table of workflow roles corresponding to the assignee. This assignee can be notified via an Asset Manager action. In order to do this, you just need to select the **Notify person or group** option (SQL name: bNotifAssignee) in the **Parameters** tab appropriately.

Note: The action notifying the assignee is triggered as soon as the task to be performed is created; that is, as soon as the transition triggering the activity is activated.

The assignee uses the **Asset lifecycle/ Infrastructure management/ My workflow tasks** navigation menu item to access the detail of the tasks to be carried out.

Note: An **executed** event is automatically created as an output event of the activity.

Example: When managing deliveries, a wizard can be used to help the user take delivery in full or in part of purchase order lines awaiting delivery.

Question type activity

These activities require the involvement of a user, specified in the **Person or group** field (SQL name: Assignee).

Question type activities are defined by:

- A question or instructions.
- Possible responses.

Examples:

- In the course of a purchase approval process, a request issued by an employee is submitted to the departmental supervisor.
- A **Question** type activity can also be used as a checkpoint to confirm that a task has been carried out. In this case, there would be, for example, one single possible response.

Specify:

1. The record in the table of workflow roles corresponding to the assignee. This assignee can be notified via an Asset Manager action. In order to do this, you just need to populate the **Person or group** field (SQL name: bNotifAssignee) in the **Parameters** tab appropriately.

Note: The action notifying the assignee is triggered as soon as the task to be performed is created; that is, as soon as the transition triggering the activity is activated.

The assignee uses the **Asset lifecycle/ Infrastructure management/ My workflow tasks** navigation menu item to access the detail of the tasks to be carried out.

2. The text of the question or the instructions to follow.
3. The possible responses. Each response is described in a sub-tab. It is identified by its description and its SQL name. To add or delete a response, right-click the sub-tab label zone and select **Add linked record** or **Delete link** from the shortcut menu.

Note: Each response automatically creates an output event for the activity.

Automatic-action type activity

These activities are carried out automatically by Asset Manager or Asset Manager Automated Process Manager.

Description

Automatic action type activities list actions to be executed.

Example: In an "Asset move" operation, an **Automatic action** type activity automatically modifies the location of all assets whose parent assets have been moved.

Indicate the list of actions to be executed here.

Note: An **executed** event is automatically created as an output event of the activity.

Execution

The workflow engine that activates the transition triggering the activity immediately executes the activity's actions. Depending on the selected options, these actions will be processed by Asset Manager Automated Process Manager or an agent in Asset Manager.

- If you select the **Execute actions immediately** option (SQL name: bExecImmediately), the workflow engine that activates the transition triggering the activity automatically executes the actions of the activity.
- Otherwise, the tasks are carried out by Asset Manager Automated Process Manager during its next verification cycle.

Test/ Script type activities

These activities are automatically performed by Asset Manager or Asset Manager Automated Process Manager.

Description

They are defined by a script and possible results.

Example: In the area of stock and purchase request management, a test/ script type activity can be used to verify that items referenced in purchase order lines are available in stock and not reserved. If this is the case, the activity can trigger a **Question**-type activity that asks the requesters if they want to reserve the item in stock.

Specify:

- The test script to be executed.
- Possible results. Each result is described in an individual sub-tab. It is identified by its description and SQL name. To add or delete a possible result, right-click the sub-tab label zone and select **Add linked record** or **Delete link** from the shortcut menu.

Note: Attention: The SQL names of each result must correspond to the return values of the test script.

Note: Each result automatically creates an output event for the activity.

Execution

The workflow engine that activates the transition triggering the activity immediately executes the activity's actions. Depending on the selected options, these actions will be processed by Asset Manager Automated Process Manager or an agent in Asset Manager.

- If you check the "Execute actions immediately option" (SQL name: bExecImmediately), the workflow engine that activates the transition triggering the activity automatically executes the actions of the activity: According to the mode of processing that you have selected for the event that triggers the transition, either Asset Manager Automated Process Manager or an agent of Asset Manager executes the actions.
- Otherwise, the tasks are carried out by Asset Manager Automated Process Manager during its next verification cycle.

Start activity

The **Start** activity is the starting point of a workflow scheme.

It is mandatory and created automatically when you create a workflow scheme. It is not possible to edit the detail.

It does not define the work to be performed.

The output events of the **Start** activity trigger the workflow instance.

Activity templates

Activity templates facilitate the creation of workflow scheme activities.

They are stored in the table of activities (SQL name: "amWfActivity").

Use the **Administration/ Workflows/ Workflow activities** templates menu item to access the list of activity templates.

Note: Warning: In order for the information contained in the detail of an activity template (activity type and so on) to be automatically copied to the level of the activities referencing this template (**Template** field (SQL name: Template) in the activity detail), the appropriate default values for the fields and links in the detail of an activity need to be defined by an Asset Manager administrator.

Triggering activities

In order for an activity to be triggered, the **Input condition** field (SQL name: selnCond) in the **General** tab of the activity detail must be populated. This condition concerns the transitions that trigger the activity.

- If there is only one transition that can trigger the activity, the transition just needs to be activated by (by Asset Manager or Asset Manager Automated Process Manager) in order for the activity to be triggered.
- If there are several transitions that can trigger the activity:
 - If the input condition of the activity is **AND**, all the transitions must be activated in order for the activity to be triggered.
 - If the input condition of the activity is **OR**, only one of the transitions needs to be activated in order for the activity to be triggered.

Note: If the input conditions of an activity are complex (combinations of **AND** and **OR**), you can create a sequence of intermediary **Test / script** type activities to achieve this.

Workflow tasks

This section explains how workflow tasks are created and executed:

Creating tasks	194
Automatic action or test/ script type activity	194
Display the list of all the workflow tasks	195
Performing a user task	195
Assigning user tasks	196
Delegating a task	196
Administrating a workflow task	197

Creating tasks

When a transition triggering an activity is activated, a task to be carried out is automatically created by the workflow engine that activated the transition.

According to the option selected in the **Log task** field (SQL name: bLogWorkItem) in the **General** tab of an activity, this task is logged to the table of workflow tasks (SQL name: WkElem).

The **Log task** option is automatically validated.

- For **Question** or **User action** type activities.
- For **Automatic action** or **Test / script** type activities, for which the **Execute actions immediately** option (SQL name: bExecImmediately) is not selected.

Caution: If a task is not logged, it is not possible to create workflow alarms associated with this task: The **Time limit** and **Alarms** tabs in the detail of an activity are not displayed if the **Log task** option is not selected.

The task is performed differently according to whether user involvement is required or not.

Automatic action or test/ script type activity

If the task results from an **Automatic action** or **Test / script** type activity whose **Execute actions immediately** option (SQL name: bExecImmediately) is selected, the task is immediately executed by the workflow engine that activated the transition which created the task. This can be either Asset Manager Automated Process Manager, or an Asset Manager agent.

Otherwise, Asset Manager Automated Process Manager verifies at regular intervals whether it needs to execute workflow task and executes them if necessary.

The frequency with which Asset Manager Automated Process Manager monitors workflow functions is defined in the options of Asset Manager Automated Process Manager.

Display the list of all the workflow tasks

To display the list of all workflow tasks, open the **Workflow tasks** (amWfWorkItem) table (**Administration/ Workflows/ Workflow tasks** link on the navigation bar).

The list that is displayed shows **all** workflow tasks, regardless of their status (pending, completed, closed), for all workflow instances.

Performing a user task

An assignee accesses their tasks via the **Asset lifecycle/ Infrastructure management/ My workflow tasks** link on the navigation bar.

More specifically, the user sees:

- Tasks that are assigned to the user and that must be completed by the user.
- The tasks that are assigned to groups to which they belong but that are not assigned to a given person.

To access the detail of the object referenced by the task, click **Detail**.

To perform the task to be carried out, display the **General** tab of the task:

- If the activity resulting from the task is a **Question** type task, the **General** tab displays the text of the question or the instructions to be followed. The possible results have corresponding buttons. Click the appropriate button. If needed, you can also enter a comment concerning the decision taken.
- If the activity is a **User action** type activity, simply click the **Wizard** button to launch the wizard to be performed.

Also, assignees can access tasks that are delegated to them via the **Asset lifecycle/ Infrastructure management/ My workflow tasks by delegation** link on the navigation bar. See [Delegating a task](#).

Assigning user tasks

The information concerning the assigning of user tasks appears in the **Assignment** tab in the detail of the task.

If you have the necessary rights, you can modify the assignment of a user task:

- Value of the **Assignment** field (SQL name: seAssignment).
- Assignee of the task.

Delegating a task

Delegations let users transmit task assignments from one user (Employee delegating) to another user (Employee delegated to) and do so for a determined or undetermined period of time.

Creating a delegation

To create a delegation, use the **Create/Modify delegations...** (sysCoreDelegation) wizard which can be accessed via the employees detail and by clicking **Delegate** (Windows client) or by selecting **Delegate** from the **Actions...** drop-down menu (Web client).

See **Common tables** guide, chapter **Departments and employees**, section **Create a delegation**.

Note: If an end date (dtEndDeleg) has not been specified for the delegation, the delegation is permanent.

Note: To enable a delegation, select the **Enable** (bActive) option in the delegation's detail.

This lets you disable temporarily a delegation without deleting it or changing its end date.

Caution: When a delegation is created, no profiles or validation rights are transferred. Consequently, employee B receiving the delegation must have the same validation rights as employee A to accept or refuse tasks that they receives by delegation.

Tip: If you are using the Windows client, you can view existing delegations via the employee's detail. Click the **Delegations** tab, then the **Preview of granted delegations** and **Preview of received delegations** sub-tabs. For additional information, read the **User interface** guide,

chapter **Planner viewer.**

Delegated tasks: Overview and viewing

When a delegation from employee A to employee B is created and enabled, all workflow tasks assigned to employee A or to a group of which employee A is a member are sent to employee A and to employee B.

Consequently:

- Employee A can still see all assigned workflow tasks via the **Asset lifecycle/ Infrastructure management/ My workflow tasks** link on the navigation bar.
- Employee B sees all assigned workflow tasks via the **Asset lifecycle/ Infrastructure management/ My workflow tasks** link on the navigation bar as well as all workflow tasks that are transmitted to them by delegation via the **Asset lifecycle/ Infrastructure management/ My workflow tasks by delegation** link on the navigation bar.

During the delegation period, employee B can view, validate or refuse all workflow tasks assigned to employee A as well as all workflow tasks assigned to a validation group of which employee A is a member.

Workflow tasks that employee B can view via the **Asset lifecycle/ Infrastructure management/ My workflow tasks by delegation** link on the navigation bar are still shown as being assigned to employee A (or to a validation group of which employee A is a member).

Administrating a workflow task

The information concerning the administration of a workflow task is contained in the **Administration** tab of the task detail.

This information is available to users with administration rights only.

Workflow events

Events are associated with activities. They trigger transitions to other activities.

There are three possible system types for an event at the level of an activity. The system type of an event is defined by the **System type** field (SQL name: seType) in the detail of the event:

- **System** event.
- **User** event.
- **Alarm** event.

This section includes:

System event	198
Alarm event	198
User event	199
General conditions of activation	201
Processing of events	202
Application: Implementing a synchronous workflow scheme	206
Terminal event	207

System event

System events are automatically defined by Asset Manager when creating/ modifying activities.

They correspond to the different possible outcomes (results) of the work carried out in an activity:

- Responses to a **Question** type activity.
- Results of a **Test / script** type activity.
- Event **executed** in the case of a **User action** or **Automatic action** type activity.

Example: If an activity asks a question for which the possible answers are "Yes" and "No", two system events are created at activity level, named "Yes" and "No".

Alarm event

Alarm events of an activity are created when you define the activity alarms that trigger events.

Such an alarm is defined in the **Alarms** tab of the activity detail. The event carries the same name as the alarm.

User event

User events are independent of the tasks carried out within an activity. They are created manually via the graphical workflow editor (**Add event** shortcut menu command).

Note: Events associated with the **Start** activity are user events.

There are two types of **User** events (**Type** field (SQL name: seMonitoringType) at the top of an event detail):

- **Database**
- **Periodical**

Database type event

Database type events allow you to activate workflow instances on specific records.

A **Database** type event occurs:

- When the general activation conditions specified in the **General** tab are fulfilled.
- When certain triggering parameters are verified at the level of the records being monitored.

Parameters tab of a Database type action

Activation conditions on monitored records:

<input type="checkbox"/> Pre-Insert	<input type="checkbox"/> Pre-Delete
<input type="checkbox"/> Post-Insert	<input type="checkbox"/> Post-Delete
<input checked="" type="checkbox"/> Pre-Update	<input checked="" type="checkbox"/> Post-Update

Fields monitored (after):

Script:

The parameters that trigger a **Database** type event are described in the **Parameters** tab of the detail of the event. The following information is specified:

- The records to be monitored (These records can be records in the table indicated in the context or linked records.) If the records to be monitored are records linked to the table indicated in the context, specify the corresponding link in the **Link / context** field (SQL name: LinkToMonitTable).
- Activation conditions for the event concerning the records being monitored. To specify the conditions of activation, you can:

Note: On the **amHistory** table, only **Pre-Delete** and **Post-Delete** conditions work.

- The **Pre-Insert** option (SQL name: bPreInsert), makes it possible to activate the transition before inserting a new record.

Caution: This option must not be used if the workflow scheme creates a record in a table AND executes an action on a record linked to this record.

For example, if the workflow scheme creates a portfolio item and modifies an asset's field that was automatically created with the portfolio item.

Use the **Post-Insert** option instead.

- The **Post-Insert** option (SQL name: bInsert), makes it possible to activate the transition after inserting a new record.

Caution: This option must not be used if the workflow scheme needs to execute a script type action to populate a field or link via the **set()** function.

Use the **Pre-Insert** option instead.

- The **Pre-Delete** option (SQL name: bDelete), makes it possible to activate the transition before deleting records.
- The **Post-Delete** option (SQL name: bPostDelete), makes it possible to activate the transition after deleting records.
- If you select the **Pre-Update** option (SQL name: bPreUpdate), you can specify the fields for which the transition will be made before these fields are updated. These fields are defined in the **Fields monitored (before)** field.

Note: If you enter * in the **Fields monitored (before)** field, updates to **any** field are monitored.

- If you select the **Post-Update** option (SQL name: bUpdate), you can specify the fields that, when modified, must be taken into account in the **Fields monitored (after)** field (SQL name: MonitFields). To specify multiple field names, use commas to separate them. If you leave the field empty, modified fields are not taken into account.

Note: If you enter * in the **Fields monitored (after)** field, updates to **any** field are monitored.

Caution: It is not possible for the activation condition of the event to be the destruction of the object referenced by the context.

- Write a script in the **Script** zone (SQL name: memScript). If you write a script and check one or more of the **Insert**, **Update** and **Delete** boxes, the script restricts the activation conditions.

Example:

If an event is to be triggered when the total price of an existing request is modified, populate the **Parameters** tab as follows:

As soon as a **Database** type event occurs, it is taken into account by the Asset Manager client machine on which it occurred. The way that it is processed depends on the option selected in the **Processing** field (SQL name: seProcessingMode) in the **General** tab of the event detail.

For further information, refer to [Processing of events](#).

Old link

The "Old" link points to the current record, monitored by an event, before one of its components is modified. Used in a script in the form [Old.<SQL name of the field>], it returns the value of the record field before modification. In this way, you can perform tests such as:

```
If [Old.lUserId] <> lUserId Then....
```

Periodical type event

Periodical type events concern a selection of records in a given table. They allow you to trigger a workflow instance on a periodical basis for each record of the selection.

Example: Every month, the residual values of assets with a "PC" nature are updated.

A **Periodical** type event occurs if the activation conditions indicated in the **General** tab are fulfilled.

In this case, Asset Manager Automated Process Manager triggers the event.

The frequency with which Asset Manager Automated Process Manager triggers **Periodical** type events is defined in the planner in the **Parameters** tab of the detail of the event.

The way that the event is processed is described in the section entitled "Processing of events", in the "Workflow" chapter of this manual.

General conditions of activation

The conditions of activation for all types of event can be defined in the **General** tab:

AQL condition (SQL name: AQLCond)

The **AQL condition** field specifies the selection of records involved in the workflow scheme.

Reinitialize workflow instance if there is already one in progress (SQL name: bReinitialize)

Note: The **Reinitialize workflow instance if there is already one in progress** option only appears in the details of events resulting from the "Start" activity.

The **Reinitialize workflow instance if there is already one in progress** option determines what happens if an output event of the **Start** activity concerns an database object that is already the subject of an instance of this workflow scheme.

What happens depends not only on this option but also on the **One single active workflow instance for an object** option (SQL name: bUniqueActive) in the **General** tab of the workflow scheme.

The following table resumes the different possible cases:

Limiting workflow instances in progress for a given object - different possible cases

		One single active workflow instance for an object option in the General tab of the workflow scheme.	
		Validated	Not validated
Reinitialize workflow instance if there is already one in progress option in the General tab of the output event of the Start activity.	Validated	If there is already a workflow instance in progress for the object, it is stopped and a new workflow instance started.	
	Not validated	If there is already a workflow instance in progress for the object, the event is ignored (no new workflow instance).	A new workflow instance is created.

Processing of events

Once the general activation conditions are fulfilled, the way that events are processed depends on:

- The event "type" (**Type** field (SQL name: seMonitoringType) at the top of an event detail).
- The option selected in the **Processing** field (SQL name: seProcessingMode) in the **General** tab if the detail of an event.

The following table resumes the different ways in which an event can be processed:

The different ways in which an event can be processed

	Log event and process by server	Log event and process immediately	Process event immediately without logging
Periodical type event	<p>Asset Manager Automated Process Manager triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the Parameters tab of the event detail.</p> <p>As soon as it occurs, Asset Manager Automated Process Manager logs the event in the table with SQL name amWfOccurEvent.</p> <p>The transition is activated later by Asset Manager Automated Process Manager (the frequency with which Asset Manager Automated Process Manager monitors the transitions to be activated is defined in the options in Asset Manager Automated Process Manager).</p>	<p>Asset Manager Automated Process Manager triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the Parameters tab of the event detail.</p> <p>As soon as it occurs, Asset Manager Automated Process Manager logs the event in the table with SQL name amWfOccurEvent.</p> <p>The transition is activated immediately by Asset Manager Automated Process Manager.</p>	<p>Asset Manager Automated Process Manager triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the Parameters tab of the event detail.</p> <p>When the event occurs, it is not logged in the table whose SQL name is amWfOccurEvent but the transition is activated immediately by Asset Manager Automated Process Manager.</p>
Database type event or system event triggered by Asset Manager (result of a Question or User action type activity, result of a Automatic action or	<p>As soon as the event occurs, it is logged in the table with SQL name amWfOccurEvent by the Asset Manager client machine.</p>	<p>As soon as the event occurs, it is logged in the table with SQL name amWfOccurEvent by the Asset Manager</p>	<p>When the event occurs, it is not logged in the table with SQL name amWfOccurEvent but the transition is activated</p>

The different ways in which an event can be processed, continued

	Log event and process by server	Log event and process immediately	Process event immediately without logging
Test / script type activity executed by Asset Manager)	The transition is activated later by Asset Manager Automated Process Manager (the frequency with which Asset Manager Automated Process Manager monitors the transitions to be activated is defined in the options in Asset Manager Automated Process Manager).	client machine. The transition is activated immediately by the Asset Manager client machine.	immediately by the Asset Manager client machine.
System event triggered by Asset Manager Automated Process Manager (result of a Test / script or Automatic action type activity executed by Asset Manager Automated Process Manager) or event on the activity alarm	As soon as the event occurs, it is logged in the table with SQL name amWfOccurEvent by Asset Manager Automated Process Manager The transition is activated later by Asset Manager Automated Process Manager (the frequency with which Asset Manager Automated Process Manager monitors the transitions to be activated is defined in the options in Asset Manager Automated Process Manager).	As soon as the event occurs, it is logged in the table with SQL name amWfOccurEvent by Asset Manager Automated Process Manager. The transition is activated immediately by Asset Manager Automated Process Manager.	When the event occurs, it is not logged in the table with SQL name amWfOccurEvent but the transition is activated immediately by Asset Manager Automated Process Manager.

The different ways in which an event can be processed

	Log event and process by server	Log event and process immediately
Periodical type event	Asset Manager Automated Process Manager triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the Parameters tab of the event detail. As soon as it occurs, Asset Manager Automated Process Manager logs the event in the table with SQL	Asset Manager Automated Process Manager triggers the event if the activation conditions are fulfilled. The frequency of triggering is defined in the Parameters tab of the event detail. As soon as it occurs, Asset Manager Automated Process Manager does not log the event in the table with SQL name amWfOccurEvent . But the

The different ways in which an event can be processed, continued

	Log event and process by server	Log event and process immediately
	<p>name amWfOccurEvent.</p> <p>The transition is immediatly activated by Asset Manager Automated Process Manager.</p>	<p>transition is activated immediately by Asset Manager Automated Process Manager.</p>
<p>Database type event or system event triggered by Asset Manager (result of a Question or User action type activity, result of a Automatic action or Test / script type activity executed by Asset Manager)</p>	<p>As soon as the event occurs, it is logged in the table with SQL name amWfOccurEvent by the Asset Manager client machine.</p> <p>The transition is activated immediately by the Asset Manager client machine.</p>	<p>When the event occurs, it is not logged in the table with SQL name amWfOccurEvent but the transition is activated immediately by the Asset Manager client machine.</p>
<p>System event triggered by Asset Manager Automated Process Manager (result of a Test / script or Automatic action type activity executed by Asset Manager Automated Process Manager) or event on the activity alarm</p>	<p>As soon as the event occurs, it is logged in the table with SQL name amWfOccurEvent by Asset Manager Automated Process Manager.</p> <p>The transition is activated immediately by Asset Manager Automated Process Manager.</p>	<p>When the event occurs, it is not logged in the table whose SQL name is amWfOccurEvent but the transition is activated immediately by Asset Manager Automated Process Manager.</p>

Using these different processing modes, it is possible to accurately specify how a workflow instance is run.

According to the selections made at the level of:

- Event types
- Event processing modes
- Activities

You can define both synchronous and asynchronous workflow schemes or combine both approaches.

Application: Implementing a synchronous workflow scheme

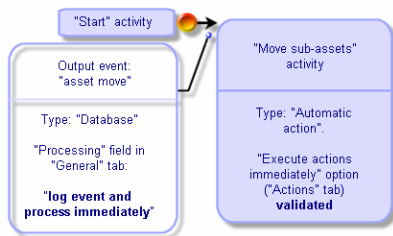
In order to accomplish a synchronous workflow scheme, you need to define:

- **Database** type event that are **Log event and process immediately** (**Processing** field (SQL name: seProcessingMode) in **General** tab of event detail).
- **Automatic action** or **Test / script** type activities, for which the **Execute actions immediately** option (SQL name: bExecImmediately) is selected, and which are triggered by these events.

Example:

Using the workflow scheme described in the diagram below, as soon as an asset changes location, its sub-assets are automatically moved to the same location:

Example of synchronous workflow scheme



In this case, when the location of an asset is modified and you click **Modify**:

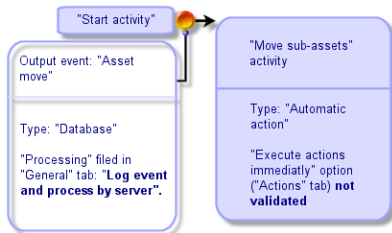
1. A database transaction starts.
2. The location of the asset is modified.
3. The workflow instance starts up.
4. The workflow transition is activated.
5. The location of the sub-assets is modified.
6. Then the full transaction is validated.

If an error occurs during one of the steps, both the locations of the asset and sub-assets are left unmodified.

If the steps are performed successfully, all the locations are modified.

On the other hand, if the same procedure is implemented using an asynchronous workflow scheme as described below, and if an error occurs, the location of the asset can be modified without the location of the sub-assets being modified.

Example of asynchronous workflow scheme



Terminal event

Definition

A terminal event ends a workflow instance, even if there are remaining tasks to be performed.

Example:

Workflow scheme with terminal event



If a workflow instance occurs as shown above and:

- The output event of activity 1 occurs and triggers activity 2, creating a task to be carried out.
- The terminal event of activity 3 occurs.

Then the workflow instance terminates, even if the task resulting from activity 2 has not been carried out.

Specifying that an event is terminal

When you create a workflow scheme via the graphical editor in the **Activities** tab of a workflow scheme, you indicate that an event is terminal as follows:

1. Right-click the event.
2. Select **Terminal event** from the shortcut menu.

Workflow transitions

Transitions link output events from an activity to other activities.

A event can be linked to several transitions.

If necessary, you can use the **AQL condition** field (SQL name: AQLCond) in the detail of a transition to specify the conditions of activation of the transition.

Workflow alarms and time limits

For each workflow activity, it is possible to define:

- A deadline for performance.
- Alarms linked to this deadline or dates stored in the database. These alarms launch actions.

Caution: If you do not activate the **Log task** option (SQL name: bLogWorkItem) in the **General** tab of the activity, you cannot define time limits or alarms.

This section includes:

Time limit	208
Workflow alarms	209

Time limit

The time limit for performance of a workflow activity is defined in the **Time limit** tab of the detail of the activity.

Note: The **Time limit** tab in the detail of an activity is only displayed if the **Log task** option (SQL name: bLogWorkItem) in the **General** tab of the detail of the activity is validated.

This time limit is defined in relation to the time at which the activity is triggered.

It is associated with a business days calendar.

You can specify a period of time or select one of the three predefined options:

- **End of next business day**
- **End of business week**
- **End of business month**

Caution: If you specify a period of time, Asset Manager considers that it is expressed in terms of business time and converts it to business hours. Example: If you enter "2 days" this is taken as meaning 48 business hours.

Workflow alarms

It is possible to associate alarms with each activity in the **Alarms** tab of the activity detail.

Note: The **Alarms** tab in the detail of an activity is only displayed if the **Log task** option (SQL name: bLogWorkItem) in the **General** tab of the detail of the activity is validated.

Time limits

The time limits that trigger alarms can be defined according to:

- A period of time after a date stored in the database (**Time elapsed since start of task** type).
- A period of time before a date stored in the database (**Time before end of task** type).
- A percentage of the time limit for performing the activity (**Time limit** field (SQL name: tsResolDelay) in the **Time limit** tab).

Note: The periods of time defining workflow time-limits are in terms of business days.

As soon as a task is created, associated workflow alarms are generated.

Workflow time limits are monitored by Asset Manager Automated Process Manager. The frequency of monitoring is defined in the options of Asset Manager Automated Process Manager.

The effect of alarms

Alarms trigger:

- Asset Manager actions.
- Or events. Events triggered by alarms are **Alarm** type events. They carry the name of the alarms that define them.

Workflow execution groups

Workflow execution groups allow you to categorize workflow schemes. The execution group to which a workflow scheme belongs is indicated in the **Execution group** field (SQL name: GroupName) in the **General** tab of the workflow detail.

Asset Manager Automated Process Manager monitors the creation of new workflow execution groups.

As soon as Asset Manager Automated Process Manager detects a new workflow execution group G, it creates a new monitoring module **Execution of workflow rules for execution group "G"**.

This mechanism is useful for the following reasons:

- It allows you to define verification timetables for each workflow execution group.
- Different workflow execution groups can be monitored by different instances Asset Manager Automated Process Manager.

Once a workflow execution group is detected, Asset Manager Automated Process Manager monitors and executes the corresponding workflow rules (monitoring alarms, processing of **Periodical** type events, activating transitions, executing tasks, and so on).

This section includes the following topic:

Priority	210
----------------	-----

Priority

For each group, you must define a priority. This priority corresponds to the order of execution of the workflow groups and enables you to define which workflow will be launched in a given period of time.

You define the priority level in the **Priority** field. A priority level 1 is more important than priority level 3.

Workflow tracking

This section includes:

Display the workflow instances of a record	211
Viewing information during different steps of the instance	212
Refreshing workflow schemes and instances	212

Display the workflow instances of a record

When a table in Asset Manager is defined as the context of the start object of a workflow scheme, a **Workflow** tab is displayed in the detail view of this table.


This **Workflow** tab shows the status of the workflow instances in progress that use this record as the start object.

- Windows client: You can display the list of the workflow instances as sub-tabs or as a list.
- Web client: Only list view is available.

Each workflow instance specifies the how the instance will progress:

- Windows client:
 - The left part lists events that have occurred.
 - The right part displays the workflow scheme. The activities to be carried out are shown as flashing.
 - The following steps are grayed out.
- Web client:
 - The top part lists events that have occurred.
 - The bottom part displays the workflow scheme. The boxes representing the activities to perform are **clear blue** and the other boxes are **dark blue**.

Viewing information during different steps of the instance

- You can view information about the steps that have been completed or that are in progress for the workflow instance (for example, to whom the task is assigned) by placing the mouse icon on the different workflow scheme boxes (an informational fly-by is displayed when you do this).
- You can display the workflow's text description and events that have been completed by clicking the  icon.

Tip: If you are using the Windows client, you can zoom in on the scheme via the zoom function.

Refreshing workflow schemes and instances

Windows client

Updating workflow schemes and instances is done dynamically: It is done automatically when the workflow scheme progresses from one step to the next or if you make modifications to the scheme itself.

Web client

There are two types of information:

- Information related to a workflow instance: Events completed, progression status (colored arrows), date on which each activity was completed, and so on. This information is updated dynamically: The graphical representation of the workflow instance is immediately updated with the changes (use the **F5** key to refresh the screen).
- Information related to a workflow scheme: list and position of boxes and links, and so on. This information is managed by a **cache** stored in memory on the Asset Manager Web Tier server and refreshed periodically (every **24 hours** by default). Each item in the cache is managed individually:

- a. When you open a workflow scheme for the first time, the workflow scheme is placed in the cache and its storage date is recorded.
- b. The next time you open this scheme, Asset Manager Web Tier checks the difference between the current date and the last storage date of the workflow scheme in the cache:
 - If the time difference is less than the pre-defined amount of time after which the cache must be refreshed, the image of the scheme is taken directly from the cache. This shortens the time needed to display the information.
 - If, however, the time difference is greater than the pre-defined amount of time after which the cache must be refreshed, the information for the workflow scheme is updated and the new storage date is saved to the cache.

To change how often the cache is refreshed, read the **Tailoring** guide, section **Customizing Web clients**, chapter **Modifying the Web client's default behavior**, section **To modify how often the cache managing the workflow schemes and instances is refreshed**.

Deleting instances of finished workflows

This section includes:

Why should you delete an instance of a finished workflow?	213
Automating the deletion of terminated workflow instances	214

Why should you delete an instance of a finished workflow?

Executing workflow schemes creates workflow instances (in the **Workflow instances** table (amWflInstance)).

These workflow instances are not automatically destroyed, even if they have finished executing (**Status** field (seStatus)).

If you use a large number of workflow schemes, you will obviously have a large number of workflow instances.

This unnecessarily increases the size of the database and can tax Asset Manager's performances.

We thus recommend that you regularly delete workflow instances that have finished executing.

Automating the deletion of terminated workflow instances

To automate the deletion of terminated workflow instances from your working database:

1. Add a field to the **Workflow schemes** table (amWfScheme) to define a deadline for when obsolete workflow instances must be deleted.
2. Create an action that deletes these obsolete workflow instances.
3. Create a workflow scheme that automates the execution of this action.
4. Configure Asset Manager Automated Process Manager to automate the execution of the workflow scheme.

Adding a field to the **Workflow schemes** table

Add the following field to the **Workflow schemes** table (amWfScheme):

Parameter	Value
SQL name	AutoCleaningDelay
Label	Instance deletion delay
Description	A deadline that, when reached, mandates the deletion of the obsolete workflow instances.
Type	Duration (time span)
Create an index for this field	Do not select this option
Description (Help tab)	Determines how much time should pass before obsolete workflow instances can be deleted.
Example	-1: Workflow instances are never deleted.Positive or null value: Workflow instances can be deleted after the deadline has passed.

To learn how to add a field to an existing table, refer to the **Tailoring** guide, chapter **Customizing the database**, section, **Creating new objects**, sub-section **Creating a field, link or index**.

Populate the **Instance deletion delay** field

Populate the **Instance deletion delay** field (AutoCleaningDelay) in each workflow scheme you use.

Connecting to the working database

1. Launch Asset Manager.
2. Connect to your working database.

Creating an action to delete obsolete instances

1. Display the list of actions via the **Administration/Actions** navigation menu.
2. Click **New**.
3. Populate the following fields:

Name	SQL name	Value
Name	Name	Delete finished instances
Context	ContextTable	Workflow schemes (amWfScheme)
Type	seActionType	Script
SQL name	SQLName	DeleteFinishedWfInstances
Script of the action	Script	See (*) below.

(*) Script of the action:

```
Const NumberOfInstanceToDelete = 50
Dim lRc As Long
Dim i As Long
i = 0
If [AutoCleaningDelay] >= 0 Then
    Dim hqWfInstance As Long
    hqWfInstance = AmQueryCreate()
    lRc = AmQueryExec(hqWfInstance, "SELECT lWfInstanceId FROM amWfInstance WHERE
lWfSchId = "& [lWfSchId] & " And seStatus = 1 AND ADDSECONDS(dtCompleted, " &
[AutoCleaningDelay] & ") < GetDate()" )
    Do While (lRc = 0 And i < NumberOfInstanceToDelete)
        Dim hrWfInstance As Long
        hrWfInstance = AmGetRecordHandle(hqWfInstance)
        lRc = AmDeleteRecord(hrWfInstance)
```

```

    lRc = AmReleaseHandle(hrWfInstance)
    lRc = AmQueryNext(hqWfInstance)
    i = i + 1
  Loop
End If


```

4. Click **Create**.

Creating a workflow scheme to automate the execution of the action

1. Display the list of workflow schemes via the **Administration/Workflow/Workflow schemes** menu.
2. Click **New**.
3. Populate the following fields:

Name	SQL name	Value
Name	Name	Delete finished workflows
Reference	Ref	ADM_CLEAN_WF_INSTC
Context of start object	StartContextTable	Workflow schemes (amWfScheme)
Execution group	GroupName	Specify any name to automate the execution of the workflow scheme in Asset Manager Automated Process Manager (ADMIN , for example).
One single active workflow instance for an object	bUniqueActive	Select this option.
Do not save instances in the database	bTransient	Do not select this option

4. Click **Create**.
5. Select the **Activities** tab.
6. Select the **Start** activity and click .
7. Populate the following fields:

Label	SQL name	Value
Name	Name	Timer
Type	seMonitoringType	Periodical
Reinitialize workflow instance if there is already one in progress	Periodical	Do not select this option
AQL condition	AQLCond	AutoCleaningDelay >= 0

8. Select and populate the **Parameters** tab according to your needs.
9. Click **Add**.
10. Right-click and select the **Add an activity** menu.
11. Populate the following fields:

Label	SQL name	Value
Name	Name	Clean W/F instances
Type	seType	Automatic action
Log task	bLogWorkItem	Select this option.
Context	ContextTable	Workflow schemes (amWfScheme)
Input condition	seInCond	OR

12. Click **Add**.
13. Select the **Parameters** tab.
14. Populate the following fields:

Label	SQL name	Value
Execute actions immediately	bExecImmediately	Select this option.
Actions	Actions	Delete finished workflows

15. Click **Close**.
16. Double-click the activity **Clean W/F instances**.
17. Right-click on the **Executed** event and select the **Detail of event** menu.
18. Populate the following fields:

Label	SQL name	Value
Processing	seProcessingMode	Log event and process immediately

19. Click **Close**.
20. Using the mouse, create a link between the **Start** and **Clean W/F instances** activities.
21. Right-click the **Executed** event and select the **Terminal event** menu.
22. Click **Modify**.

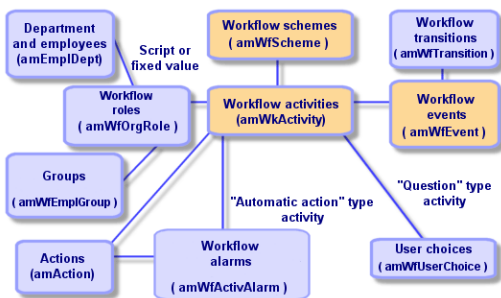
Configuring Asset Manager Automated Process Manager to automate the execution of the workflow scheme

1. Launch Asset Manager Automated Process Manager.
2. Configure the module that will trigger the execution of the **Delete finished workflows** workflow via the **Tools/Configure modules** menu.
It is the module named **Execute workflow rules for execution group 'X'**, where **X** is the value of the **Execution group** field (GroupName) defined at the level of the workflow scheme.
3. Leave Asset Manager Automated Process Manager active if you want the workflow to execute automatically.

Technical information: Data model

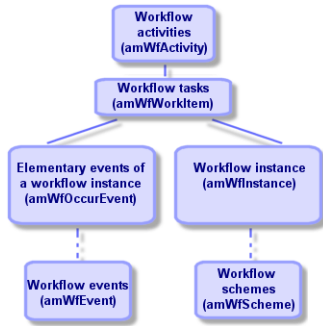
The following diagram presents the main tables involved in workflow and the links between them. The tables are identified by their label and SQL name:

Workflow - main tables allowing you to define a workflow scheme



The following diagram presents the main tables involved when a workflow instance is in progress:

Workflow - main tables involved in a workflow instance being executed



Chapter 9: Exporting data and creating SQL views

This chapter explains how to export Asset Manager data and manage SQL views of the database and includes:

Definitions	220
Exporting data from the Asset Manager database	221
Managing SQL views in the Asset Manager database	223
Recommendations	223
Defining an export script	224
Executing an export script	229

Definitions

This section includes:

Export scripts	220
Export queries	221

Export scripts

The export scripts enable you to export data or (re)create/delete SQL views via Asset Manager Export Tool or **amexp.exe**. You can save the export scripts that you define to reuse them.

An export script runs either:

- in "Export Mode" for exporting data.
- or in "Views Mode" to (re)create or delete SQL views from the database.
- Actions to perform, for creating/deleting SQL views.

An export script runs either:

- In "Export data" mode for exporting data.
- Or in "Create/Drop SQL views" mode to (re)create or delete SQL views from the database.

Export queries

You define export queries using Asset Manager Export Tool.

An export query is defined by:

- A name.
- An eventual export file (when using the "Export data" mode).
- A comment (which is not exported).
- A starting table.
- A list of columns to be extracted (fields, links, features and calculated fields from the starting table) and associated sort criteria.
- A filter containing the WHERE clause and defining the extraction conditions.
- A filter containing the HAVING clause and defining the extraction conditions.
- The text of the query (corresponding to the **Filter (WHERE clause)** and "HAVING Clause") tabs.
- A preview tab.

Exporting data from the Asset Manager database

You can export data from the Asset Manager database to text files:

- Using an export script.
- Via the **Utilities/ Export the list** shortcut menu item. This menu item is displayed when at least one list or tab list is displayed. It allows you to export the active list.


Note: In Windows Vista, Windows 8, Windows 8.1, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2, Asset Manager Export Tool requires the administrator permission to modify some local files. Therefore, it is recommended that you use the **Run as administrator** option to start Asset Manager Export Tool.

You can also find more information about extended exporting functions from the *Exporting Extended Database Structure* white paper that is included in the Asset Manager installation directory.

This section includes:

Exporting data using an export script	222
Exporting data using the Export the list shortcut menu item	222

Exporting data using an export script

1. Start Asset Manager Export Tool using the **Start** menu or from the Asset Manager program group.
2. Define an export script whose mode is set to "Export data" mode:
 - a. In the **Queries** tab, write the queries defining the data to export.
 - b. In the **Formatting** tab, specify the format for the text files to which the data will be exported.
 - c. Use the **File/ Save script** or **File/ Save script as** menu item to save the script.
3. Execute the export script:
 - o Either directly in Asset Manager Export Tool using the **Execute script** button 
 - o Or by launching **amexpl.exe** in DOS.

Note: To preserve the coherency of the access restrictions that you defined in Asset Manager, you can only launch Asset Manager Export Tool or **amexpl.exe** as an administrator ("Admin" or user with administrative rights).

Exporting data using the **Export the list** shortcut menu item

The **Export the list** shortcut menu item is accessible to all Asset Manager users; it allows users to export the data they are authorized to view.

1. Display the list you want to export (main list or tab list). If several lists are displayed on the screen, make sure you are in the list to export.
2. Select the **Export the list** contextual menu item.
3. Populate the window that appears, then click the **Export** button.

Note: For further information on the **File/ Export** menu item, refer to the **User interface** guide, chapter "First steps with Asset Manager", section "Record lists", sub-section "Exporting a list".

Managing SQL views in the Asset Manager database

Asset Manager Export Tool allows you to create, recreate or delete SQL views in the Asset Manager database. External tools can then use these views instead of text files.

Note: Warning: The SQL views that the export scripts allow you to create/modify/delete are different from views in the Asset Manager sense of the term. A SQL view is equivalent to the SQL "CREATE VIEW" statement.

To create, recreate or delete SQL views in the Asset Manager database:

1. Start Asset Manager Export Tool.
2. Define an export script whose mode is set to "Create/Drop SQL views":
 - a. In the **Queries** tab, write the queries defining the data to extract.
 - b. In the **Views** tab, specify the actions to perform: create, modify or delete views, directly execute the resulting SQL script, or save to a file.
 - c. Save the export script.
3. Execute the export script.
 - o Either directly in Asset Manager Export Tool.
 - o Or by launching **amexpl.exe**.

Recommendations

We advise against using the "Id" fields of tables as reconciliation keys if you want to reimport data you have exported. In effect, the corresponding identification numbers are not fixed and can be subject to modification. Use keys whose values are "changeless" such as the AssetTag of assets.

Defining an export script

To export data or generate SQL views for your database, you must define the export scripts and export queries they contain. To do this, use Asset Manager Export Tool.

An export script runs either:

- In "Export data" mode for exporting data.
- Or in "Create/Drop SQL views" mode to (re)create or delete SQL views from the database.

This section explains how to create export scripts:

Methodology	224
Defining export queries	225
Output format of an export script	227
Actions concerning SQL views	228

Methodology

To create or modify an export script:

1. Start Asset Manager Export Tool.
2. Open the appropriate database. Warning: You can only connect using the "Admin" login or a login with administration rights.
3. Create a new script using the **File/ New script** menu item, or open a script to be modified using the **File/ Open script** menu item.
4. At the top of the Asset Manager Export Tool screen, define if you want to export data (Export data mode) or manage SQL views (Views mode) of the database.
5. Write the queries of the export script in the **Queries** tab.
6. If you export data, specify the output format for the exported data in the **Formatting** tab.
7. If you want to manage SQL views, specify what you want to do in the **Views** tab.
8. Save the script using the **File/ Save script** menu item or **File/ Save script as**.

Defining export queries

You can define the queries of an export script in the **Queries** tab of Asset Manager Export Tool.

- Click the **New** button to add an export query.
- Click the **Delete** button to delete a selected export query.


Create a query in an export script

1. Click **New** in the **Queries** tab.
2. Define the query name. This name is used in the execution log in the **Messages** tab of the export script detail.
3. You can enter comments (they are not exported).
4. Define the data to be extracted, in the form of a script, in the **Query** field.
5. If you want to export data rather than create/modify/delete views, then in the **File** field specify the path and name of the output text file to which the exported data selected by the query will be written. Thus an export script containing several export queries will generate several text files.

Note: The **File** field is not displayed if you selected the **Create/Drop SQL views** option.

Data to be extracted

To indicate the data to be extracted, populate the **Query** field in the detail of the export script query. The query applies to a table in the Asset Manager database.

You can enter the query directly, or click the  button to access a window that will help you define the query:

Columns to be exported and sort order

In the **Columns and sort** tab (detail of the query), you define the list of fields, links, features and calculated fields to be exported, as well as the associated sort criteria.

Select one-by-one all the fields, links, features and calculated fields used for the export from the list on the left, and click the arrow to insert them in the list on the right.

For each line in the list on the right:

- Select the **Visibility** check box to export the column. If the **Visibility** box is not checked, the column is not exported (it may be used, however, to sort exported data, and so on).
- Select the **Group by** check box to group data by the field corresponding to the column. This is equivalent to adding the "GROUP BY <Field name>" clause to your SQL query.

For example:

```
SELECT Brand, Count(1ModelId) FROM amModel GROUP BY Brand ORDER BY Brand
```

Caution: If you check the **Group by** box, the "GROUP BY" is appended to the query, but in order for the query to be valid, you must also add the appropriate aggregate functions in the SELECT clause.

Define the sort order for the exported data:

1. In order to define a sort by index, select an index in the **Sort by index** field.
2. Otherwise, check the appropriate **Sort** boxes in the desired sort order.

Note: You can select the **Force indexes** option to force the use of indexes specified in the query.

For further information, refer to chapter **AQL queries**, section **Sorts and indexes** of this guide.

If you check the **Unique records only** option, lines that are absolutely identical are only exported once. This is equivalent to adding the "DISTINCT" clause to the SQL query.

Example with no check in the **Unique records only** box:

```
SELECT Brand FROM amModel
```

Example with a check in the **Unique records only** box:

```
SELECT DISTINCT Brand FROM amModel
```

Filters

You can define two types of filters for selecting the data to be extracted:


- An AQL query using the WHERE clause in the **Filter (WHERE clause)** tab.
- An AQL query using the HAVING clause in the **HAVING Clause** tab.

Displaying the query

The AQL query that you define in the **Columns and sort**, **Filter (WHERE clause)** and **HAVING Clause** tabs is displayed in the **Queries** tab.

Previewing the query results

You can test the query and view it in SQL language syntax from the **Preview** tab.

Simply click  to preview the query results as a list of records. Note that Asset Manager displays the number of records matching your query at the bottom right of the window.

Output format of an export script

If you select the **Export data** function, you can define the format for the output text files in the **Formatting** tab. This format is applied to all export queries.

Note: The **Formatting** tab is not displayed if you choose to drop (delete)/ create/ recreate views.

Column title

Select a value if you want the first line of the export file to include:

- The alias of the columns specified in the export query.
- The "SQL name" of fields or links corresponding to columns.
- The "Description" of fields or links corresponding to columns.

Column separator

This separator is inserted between the data in each column.

Text identifier

The identifier surrounds text strings. If you use the ' character, any ' characters you export will be output as ". And vice-versa for the " character.

Character set

This option allows you to choose either the ANSI, OEM(DOS), UFT-8, UNICODE or Latin 1. character set.

Decimal separator

This character is used to separate the decimal part of exported numbers.

Date separator

This character is inserted between the day, the month and the year of exported dates.

Date format

The date format defines the order in which days (DD), months (MM) and years (YY) are exported.

Year format

Defines whether years will be exported with 2 or 4 digits.

Time separator

This character is inserted between the hours, minutes and seconds.

Display seconds

Indicate if you want seconds to appear in exported times.

Actions concerning SQL views

If you want to delete or (re)create SQL views corresponding to export queries, you can use the **Views** tab to define the actions to be performed.

Note: The **Views** tab is not displayed if you choose the **Export data** option.


Select one of the actions to perform in the "Actions" frame:

- Create or recreate views.
- Delete views.

In the "SQL view-manipulation script" section, specify how you want the query to be processed (**Queries** tab, **Actions** frame):

- To create or recreate SQL views directly when executing the export script, select the **Execute SQL directly** option.
- To generate a SQL view script to create a view ("CREATE VIEW" statement) or drop a view

("DROP VIEW" statement), select the **Save the SQL code to a file** option, then:

- a. Click the  button to indicate the name and path of the file.
- b. Specify a SQL statement separator: ";" (for Oracle) or "GO" (for all other DBMSs).

Executing an export script


You can use export scripts to export data or manage SQL views.

This section describes two methods for executing an export script:

Executing an export script from Asset Manager Export Tool	229
Executing an export script from DOS	230

Executing an export script from Asset Manager Export Tool

To execute an export script from Asset Manager Export Tool:

1. Start Asset Manager Export Tool.
2. Define your export script and save it.
3. Then execute the script:
 - o Either by using the **Actions/ Execute the script** menu item.
 - o Or by pressing F8.
 - o Or by clicking .

Information about the progress of the export process is displayed in the **Messages** tab.

If the export process terminated successfully, the last message displayed is: "Script successfully executed ". If an error occurs, the following message is displayed: "An error occurred while executing script".

An icon is displayed before all messages:

 General information.

 Error.

• Export was successful.

⚠ Warning.

Executing an export script from DOS

How it works

In order to execute the DOS software "online", you must first create an export script using Asset Manager Export Tool.

You can execute, either manually or automatically (with a batch file, for example), an export command using the **amexp.exe** program in the **bin** sub-folder of the Asset Manager installation folder.

Syntax

```
amexpl [-verbose] [-?|h|H] -script:<script>  
-cnx:<cnx> [-login:<login>]  
[-password:<password>]
```

-verbose: displays messages during the export process.

-, -h or -H: displays help messages.

-script: path and name of the export file to execute.

-cnx: name of the connection to the Asset Manager database (as it appears in the **File/ Manage connections** menu item).

-login: login name of the database administrator ("Admin" or a user with administrative rights).

-password: password for the login.

Strings between <> cannot include spaces.

For example:

```
amexpl32 -verbose -script:ibmassets.scx -cnx:GeneralDatabase -login:Gerald -  
password:PAssword
```

Chapter 10: Scripts

This chapter explains how to use scripts and includes:

Definition of a script	231
Applications of scripts	233
Introduction to functions	234
Classifying Basic functions	238
First steps in writing scripts	238
Script libraries	241
Tips and warnings	243
First example	249
Second example	251

Definition of a script

This section includes:

Overview	231
Information about this version of Basic	232
Data access notation	232

Overview

The word "script" generally designates a program written in a high-level language. In Asset Manager, this notion includes three types of scripts:

- **Procedural scripts**, which include:
 - Calculation scripts written in Basic used to calculate the values of fields, determine the properties of objects in the Asset Manager database, and so on.

- Basic scripts executing tasks, particularly in actions.

Note: These Basic programs can incorporate functions. This type of script is the subject of this chapter.

- **Declarative scripts.** These are import and export scripts that use their own scripting language, different from Basic. This type of script is documented in full detail in the **Administration** manual, chapter **Importing data** and in the **Exporting data and managing SQL views** chapter of this manual.
- **"Mixed"**, both declarative and procedural. This type of script is used in the wizards in Asset Manager.

Information about this version of Basic

The version of Basic used in Asset Manager is a sub-set developed by Cypress compatible with "Visual Basic for Applications™". Refer to the Basic documentation for additional information on this language, its structure and syntax.

Only certain **Visual Basic for Applications** functions are supported, for example:

- File access functions are not supported.
- There is limited support for date and time functions.
This is particularly true in Linux.
- The **Visual Basic for Applications** controls are not available.

Note: To consult the Programmer's Reference of a Basic function or keyword, position the cursor on the word and press F1: Contextual help is displayed.

Data access notation

The Basic syntax used in Asset Manager is similar to standard syntax, except for data access functions from the current record; this uses the following format:

[Link.Link.Field]

Example from the Models table:

[Category.FullName]

Note: You can use the following syntax to recover the ID number of a link:

```
[Link.Link]
```

When you want to refer to a link, you can use either the link's SQL name or the link's key name.

Example:

```
RetVal=[Contact.Location] or RetVal=[Contact.lLocaId]
```

Both examples return the same result, the ID of the link.

Applications of scripts

Asset Manager allows you to associate the following properties with a Basic "Script":

- For configuring the default values of fields (**Configure object** command in the shortcut menu).
- For the default value of a feature associated with a table.
- In Basic type calculated fields.
- For configuring fields (**Configure object** command in the shortcut menu or Asset Manager Application Designer):
 - **Default value.**
 - **Mandatory nature.**
 - **History.**
 - **Read-only.**
- For the parameters of a feature associated with a table:
 - **Default value** (SQL name: DefValScript).
 - **Available** (SQL name: seAvailable).
 - **Force display** (SQL name: seForceDisplay).
 - **Mandatory** (SQL name: seMandatory).
 - **Keep history** (SQL name: seKeepHistory).
- For **Script** actions:
 - **Action script** (SQL name: Script) for a **Script** action.
- In the wizards:

- Start and end of wizard scripts.
- Scripts for defining the node properties.
- In "Basic" calculated fields.
- In the workflow:
 - For **Test / script** workflow activities.
 - For **Base** workflow events.
 - For **Calculated individual** workflow assignees in the amWfOrghole table.

Introduction to functions

This section includes:

Definition of a function	234
Built-in functions and programmable functions	235
Function and parameter types	236

Definition of a function

A function is a program that performs operations and returns a value to the user. This value is named the "return value" or "return code".

Functions have the following structure:

```
Function <Function name> (<Parameter> As <Parameter type>[, ..., <Parameter> As  
<Parameter type>]) As <Function type>
```

```
<Program (script) executed by the function. This program must define the return  
value.>
```

```
End Function
```

```
End Function
```

This structure applies to both built-in functions and programmable functions.

Built-in functions and programmable functions

Built-in functions and programmable functions are the two major function categories available under Asset Manager.

Built-in functions

Built-in functions are similar to software items that have already been written for the user. These software items perform all types of tasks (calculations, conversions of data provided by the user) and return a result. The user simply calls the function by its name and provides any information required to return a result. The items of information provided by the user are named the "parameters".

For example, the **AmConvertCurrency()** function converts an amount in currency A to an amount in currency B, using an exchange rate defined at a given date. In this example:

- The function name is `AmConvertCurrency`
- The parameters the user must provide are:
 - Currency A
 - Currency B
 - The amount to convert
 - The date when the conversion will take place (used to identify the conversion rate to use).


This function performs the conversion, and provides a return value corresponding to the result of the conversion.

Programmable functions

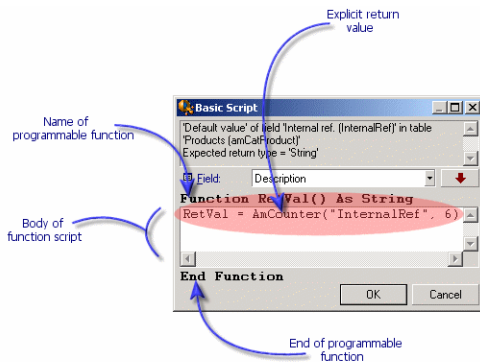
Programmable functions are software items users can write themselves. The user is responsible for explicitly defining the value to be returned in the **RetVal** variable (also named the "return value") by the programmable function, in the following format:

```
RetVal=<Expression>
```

Note: Asset Manager refuses to compile the script of a function for which the return value is not defined.

Programmable functions are accessible through a script builder (by clicking the  button in a scriptable field). The script builder is designed to help users create the software item corresponding to a function. The script builder includes a template for writing programmable functions:

Script - builder



A description the programmable function is available at the top of the script builder window. It identifies the object concerned by the function (for example, the default value of the **Bar code** field (SQL name: BarCode) in the table of assets) as well as the expected return type code (using the previous example: "String").

Function and parameter types

Function types

The type of a built-in function is the type of the value returned by the function.

Note: We suggest you pay special attention to this point, because it can cause compilation and execution errors in Basic scripts.

For example, you cannot use a function that returns a value of one type when defining the default value of a field of a different type. For example, try to assign this default script to any "Date" or "Date and time" type field:

```
RetVal=AmLoginName()
```

The `AmLoginName()` function returns the name of the connected user, in the form of a character string (String type). This return value is therefore in a format incompatible with the format of a "Date" field, and Asset Manager will display an error message the next time that you create a record in the same table.

Parameter types

The parameters used in built-in functions also have a type; you must respect that type for the function to execute correctly. If an error occurs in the type of a parameter, Asset Manager displays an error message when executing the function.

List of types

The following table summarizes the various types available for a function or a parameter:

Functions/parameters - types

Type	Description
Integer	Integer from -2,147,483,648 to +2,147,483,647.
Long	Integer from -2,147,483,648 to +2,147,483,647.
Double	8 byte floating-point number.
String	Text in which all characters are allowed.
Date	Date or Date+Time.
Variant	Generic type that can represent any type.

Determining the return type of a programmable function

Before editing a script, you should determine the function involved and its type. This information is displayed in bold in all "Basic script" windows in the following format:

```
Function <Function name>() As <Function type>
```

The three most common function types are "Boolean", "Integer" and "String":

- "Boolean" functions return either "TRUE" or "FALSE"; any other value causes an error when compiling the Basic script.
- "Integer" functions return only integer values (for example. 0, 1, 8, 12).
- "String" functions return only character strings (for example: "Building21") between quotes.

Note: If you do not respect the type of a function, you may obtain errors when compiling the Basic program. Always note the type of the function you are using.

The function name and type allow you to determine the return code you must use in the script, in the following format:

RetVal=<Expression respecting the function type>

Classifying Basic functions

The Basic used in scripts uses various classes of functions:

- Traditional Basic functions following the "Visual Basic for Applications TM" standard
- Generic functions specific to Asset Manager, which may be used in all places where scripts are used.
- Specific functions, which may be used in certain parts of Asset Manager.

First steps in writing scripts

This section deals with the functioning of scripts and includes an example scenario:

Example scenario	238
Step 1: Create the feature "Tutorial"	239
Step 2: Open the edit window	239
Step 3: Analyze and define the algorithm	239
Step 4: Draw up the Basic script	240
Step 5: Test the Basic script	240

Example scenario

Objective

To make sure that the "Tutorial" feature is only available for the "Computer/ Motherboard/" model and its descendants.

Method


Attaching a Basic script to the **Available** parameter (SQL name: seAvailable) of the "Tutorial" feature.

Step 1: Create the feature "Tutorial"

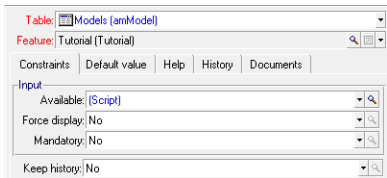
Select the **Administration/ Features/ Features** navigation menu item. Click **New** to create a new feature. Populate this feature as shown below:

Field name	Value
Title (SQL name: TextLabel)	"Tutorial"
SQL name (SQL name: SQLName)	"Tutorial"
Input type (SQL name: seDataType)	Numerical


Click **Create** (Windows client) or **Save** (Web client) to create the feature.

Go to the **Parameters** tab and click  (Windows client) to edit the parameters of the "Tutorial" feature. Populate the **Constraints** tab as shown below.

Note: Currently, you cannot modify the **Constraints** tab in the Web client.



Step 2: Open the edit window

In the **Constraints** tab, set the **Available** parameter (SQL name: seAvailable) to "(Script)". Click the magnifier button . Asset Manager opens the script edit window:

Step 3: Analyze and define the algorithm

The algorithm must fulfill the following tasks:


- Set the **Available** field (SQL name: seAvailable) to **Yes** if the model is `"/Computer/ Motherboard/"` or one of its descendants.


- Set the **Available** field to **No** in all other cases.

Our algorithm is therefore as follows

```
If the model's full name starts with "/Computer/Motherboard" Then  
The feature is available  
Otherwise  
The feature is not available
```

It is therefore the value of the **Full name** field (Nom SQL : FullName) in the Models table that determines the value of the **Available** field of the feature. Only this field appears in our algorithm.

Click the magnifier button  next to the **Available** field to start editing the Basic script. The drop-down list in the edit window lets you select the **Full name** field (SQL name: FullName) from the Models table.

Once you have selected the field, transfer it to the edit window by clicking the  button.

Step 4: Draw up the Basic script

You now need to write the algorithm from step 3 in Basic in the edit window established for this task.

```
If Left([FullName], Len("/Computer/Motherboard/"))="/Computer/Motherboard/" Then  
    RetVal=1  
Else  
    RetVal=0  
End If
```



Note: Scripts are not case-sensitive.

Click **OK** to confirm your script.

Step 5: Test the Basic script

This step allows you to make sure that the script functions correctly.

1. Open the **Models** table by selecting the **Portfolio management/ Asset configurations/ Models** navigation menu. Click **New** to create a new model.
2. Populate the mandatory fields only.
 - a. **Name**
 - b. **Sub-model of** (SQL name: Parent) with "Computer/Motherboard".

- c. **Nature** (SQL name: Nature)
 - d. **Bar code** (SQL name: BarCode).
3. Click **Create** (Windows client) or **Save** (Web client) to create your new model.
 4. Select the **Features** tab and click  to add a feature. The selection screen shows the name of the feature for which you have just edited the script.
Note: Currently, you can only add the feature in the Windows client.
 5. Change the value of the **Sub-model of** field to "/Computer/" and validate this change by clicking **Modify** (Windows client) or **Save** (Web client).
 6. Select the **Features** tab in the model detail and click the  button to add a feature. The selection screen no longer shows the name of the feature for which you have just edited the script.

The script fulfills its function correctly.

Script libraries

Asset Manager enables you to save script libraries in order to centralize the access to these scripts.

You can open the script library via the **Administration/ Scripts** navigation menu.

The recorded script libraries are called up by the **amEvalScript** API command.

For more information about the **amEvalScript** API, refer to the **Programmer's Reference** guide, section "Alphabetic reference".

This section includes:

Concepts	241
Creating a script library	242
Calling up a script in a script library	242

Concepts

In Asset Manager, a script defines a function.

Creating a script library thus implies that you will define a set of functions.

Creating a script library

To create a script library:

1. Open the list of script libraries.
2. Click **New**.
3. Populate the **Name** field with the name of your library.
4. Enter your script in the **Script** field.
5. Confirm your script by clicking **Create** (Windows client) or **Save** (Web client).

Note: Windows client validates the script and prompts you with error message when there are syntax errors; while Web client does not.

For example, create the "lib" library by entering the following script:

```
function FullName(strName As String, strFirstName As String) As String
    FullName = strFirstName & ", " & strName
end function
```

This function returns a string composed of an employee's first and last name.

Caution: Each created function must have a different name for the set of script libraries created.

Calling up a script in a script library

To call up a script from a library, you must define several parameters: the name of the library, the function defined in the script, and the parameters associated with this function.

For example, create a script-type action, "callEvalScript", that will use the previously created library.

1. Populate the **Context** field (SQL name: ContextTable) with the **Employees and departments** table (SQL name: amEmplDept).
2. Enter the following script in the **Script** tab:

```
Dim strFullName As String
strFullName = amEvalScript("biblio", "FullName", "", [Name], [FirstName])
amMsgBox (strFullName)
```

3. This script calls up the "FullName function from the "lib" library and displays the first and last name of the employee in the dialog box.
4. Confirm your script by clicking **Create** (Windows client) or **Save** (Web client).

Note: Windows client validates the script and prompts you with error message when there are syntax errors; while Web client does not.

Note: The context parameter, normally used with the **amEvalScript** API is not used in the case where you call on a script library.

Tips and warnings

This section includes several tips that may help you when writing scripts.

Precautions when using programmable functions	243
Format of "Date+Time" constants in scripts	244
Format of "Duration" constants in scripts	244
Read and write access to a system itemized list	245
CurrentUser virtual link	246
Old virtual link	247
Commenting a Basic script	247
Triggering an error message	248
Using recursive functions	248
Force Unicode text file	248

Precautions when using programmable functions

Here a few precautionary measures you should remember when writing your scripts:

- The purpose of programmable functions, such as that which defines the default value of a field or a link, is to set the return value of the function. Therefore you are strongly advised against performing other operations within a programmable function. In the best case scenario, you may note a general performance hit, and in the worst case, you can damage your database.

- Programmable functions are widely used in Asset Manager. When possible, try to optimize your scripts to maintain overall Asset Manager performance.

Format of "Date+Time" constants in scripts

Dates referenced in scripts are expressed in international format, regardless of the display options defined by the user:

yyyy/mm/dd hh:mm:ss

Example:

```
RetVal="2001/07/12 13:05:00"
```

Note: You can also use a hyphen ("-") as a date separator.

"Basic" date

In Basic, a date can be expressed in international format, or as a "Double-precision number". In this case, the integer part of the number represents the number of days elapsed since December 30, 1899 at midnight, the decimal part represents the fraction of the current date (The number of seconds elapsed since the start of the day divided by 86400).

"Unix" date

Dates are expressed differently in Basic and under Unix:

In Unix, dates are expressed as an "Integer (32-bit)" that represents the number of seconds elapsed since January 1, 1870 at midnight, independent of time zones (UTC time).

Format of "Duration" constants in scripts

In scripts, durations are stored and expressed in seconds. For example, to set the default value for a "Duration" type field to 3 days, use the following script:

```
RetVal=259200
```

Likewise, functions that calculate durations, such as the `AmWorkTimeSpanBetween()` function, return a number of seconds.

Note: For conversions, Asset Manager considers a year as being 12 months and a month being 30 days (thus 1 year = 360 days).

Read and write access to a system itemized list

Asset Manager manages system itemized lists by assigning an integer to each possible value in the itemized list.

Let's take the example of the itemized list used to populate the **Assignment** field (SQL name: seAssignment) in the **Standard assignment** zone of the **General** tab of an asset detail.

The following table summarizes the values used in this itemized list:

Value in itemized list	Associated integer
In use	0
In stock	1
Retired (or consumed)	2
Awaiting delivery	3

Thus, to define the default value of an itemized list, you need to:

1. Identify the integer corresponding to the appropriate value
2. Edit the following string:

```
RetVal=<Integer associated with the appropriate value>
```

Using this example, if you want to set the default value of the system itemized list used in the **Assignment** field to **Awaiting delivery**, you need to edit the string as follows:

```
RetVal=3
```

Note: Do not confuse a system itemized list with a user-defined closed itemized list.

Note: The full list of system itemized list values is included in **database.txt**, which can be found in the **doc\infos** sub-folder of the Asset Manager installation folder. The two columns, "Data display and entry type" and "Additional information on data display and entry type", describe the itemized list type and the values taken by an itemized list respectively.

CurrentUser virtual link

Definition

"CurrentUser" may be considered as a link starting in all tables and pointing to the record in the table of departments and employees corresponding to the current user.

- In the "CurrentUser" format, it points to the record corresponding to the current user, and returns the description string from the Employees and departments table.
- In the "CurrentUser.<SQL name of field>" format, it returns the value of the field for the current user.

Note: This virtual link is not displayed in the list of fields and links; therefore it is not directly accessible in the script builder. You must enter this expression manually.

Equivalencies

The `AmLoginName()` and `AmLoginId()` functions, which return the current user's name and ID, respectively, may be considered as functions derived from "CurrentUser". In effect, the following are equivalent:

- `AmLoginName()=[CurrentUser.Name]`
- `AmLoginId()=[CurrentUser.IPersId]`

Restrictions

`CurrentUser` will only work if a context is defined (the context being a table).

If there is no context, you must use another function.

Example:

You want to create a non-contextual action to execute a file whose path depends on the user connected to the Asset Manager database.

If the action was contextual, you could have created an **Executable** type action with the **Folder** field (Folder) set to, for example: `c:\scripts\[CurrentUser.Name]`.

However, when an **Executable** type action does not have a context, `[CurrentUser.Name]` is considered as fixed text.

You must therefore find another solution, such as creating a **Script** type non-contextual action as follows:

```
RetVal = amActionExec("program.exe","c:\scripts\" + amLoginName())
```

Old virtual link

This link is used in the scripts belonging to actions used in a workflow.

Definition

The Old link makes it possible to reference the context of the workflow before modifying the record.

Restriction

This link may only be used in scripts belonging to actions used by a workflow.

Syntax

To obtain the value of the field before updating the field by the workflow, use the following syntax:

```
[Old.Field]
```

Commenting a Basic script

It is often useful to comment a Basic script to specify, in clear terms, what it performs or to allow a user to understand and to be able to modify the script. Asset Manager provides you with the ability to comment the body of a script using the apostrophe (') character. All the characters following a single straight quote mark on the same line are ignored by the compiler, which interprets them as comments. There are two possible situations:

- Either the comment has its own line in the Basic script, as shown below.
- ```
' Here we test the value of the Bar code field in the table of assets
' If this value is PC1, the return code is set to TRUE
If [BarCode]="PC1" Then
 RetVal=True
End If
```
- Or the comment is added to the end of a line that must be interpreted by the Basic compiler.

```
If [BarCode]="PC1" Then ' Then BarCode is PC1
RetVal=TRUE ' The return value is set to TRUE
End If ' End of test
```

## Triggering an error message

You can trigger an error message on purpose using the Err.Raise function. Its syntax is as follows:

```
Err.Raise (<Error number>, <Error message>, [Hide system error])
```

- If **[Hide system error]** is set to '1', then the function hides the unwanted system error message (the 'error in Line number' information).
- If it is set to '0', the function returns the full information.

**Note:** When the creation or modification of a record is invalidated because of the value of the "Validity" field of the table concerned, it is good practice to trigger an error message with the Err.Raise function, in order to warn the user. Without this error message, the user will not necessarily understand why the record cannot be created or modified.

## Using recursive functions

In the scenario that a recursive function in Asset Manager uses local variables, the result of the function may be incorrect. This is a known limitation that Asset Manager scripts only support tail recursion (also known as tail call) in this particular scenario.

## Force Unicode text file

By default, scripts are written with the ANSI character set. You can change it to Unicode by following these steps:

1. Open the Asset Manager Windows Client.
2. On the **Edit** menu, click **Options**.
3. Expand the **Advanced** mode.
4. Change the **Force Unicode text file** option from **No** to **Yes**.



## First example

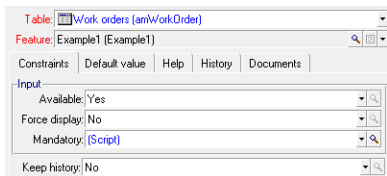
This section deals with a hypothetical problem that can be solved using a Basic script. We recommend trying this problem yourself before consulting the described solution.

This section includes:

|                                                |     |
|------------------------------------------------|-----|
| Description of the problem .....               | 249 |
| Step 1: Analyze and define the algorithm ..... | 249 |
| Step 2: Draw up the Basic script .....         | 250 |
| Step 3: Test the Basic script .....            | 251 |

## Description of the problem

A feature named "Example1", associated with the table of work orders, must be populated when work orders are closed. This feature is optional for work orders that are not closed. The rest of this example supposes that the feature has already been created, has an arbitrary input type, is associated with the table of work orders, is available and is displayed by default (force display) as shown in the screen capture below:



## Step 1: Analyze and define the algorithm

The algorithm must fulfill the following tasks:

- Set the **Mandatory** field (SQL name: seMandatory) to **Yes** if the ticket is closed.
- Set the **Mandatory** field (SQL name: seMandatory) to **No** in all other cases.


Our algorithm is therefore as follows:

```
If the work order is closed Then
 Populating the feature is mandatory
Else
 Populating the feature is not mandatory
```

A work order is closed if its **Status** field (SQL name: seStatus) is either **Closed**.

It is therefore the value of the **Status** field (SQL name: seStatus) in the table of work orders that determines the value of the **Mandatory** field (SQL name: seMandatory) of the feature. Only this field appears in our algorithm.

The drop-down list in the edit window allows you to find the **Status** field in the table of work orders.

Once you have selected the field, transfer it to the edit window by clicking the  button.

This field is populated using a system itemized list. We therefore have:

| Value in itemized list | Associated integer |
|------------------------|--------------------|
| <b>Notified</b>        | 0                  |
| <b>Scheduled</b>       | 1                  |
| <b>In progress</b>     | 2                  |
| <b>Closed</b>          | 3                  |

The value in the itemized list that interests us is therefore:

- **Closed** corresponding to the value "3"

## Step 2: Draw up the Basic script


```
If [seStatus]= 3 Then
 RetVal = 1
Else
 RetVal = 0
End If
```

You now need to write the algorithm from step 1 in Basic.

Click **OK** to confirm your script.

## Step 3: Test the Basic script

This step allows you to make sure that the script functions correctly.

1. Open the table of work orders by selecting the **Asset lifecycle/ Work orders/ Work orders** menu item. Select a ticket that is **Closed** (or create a work order with this status if there isn't one available).
2. Select the **Features** tab. Asset Manager has added the feature to the work order concerned and the feature is mandatory.
3. Now select a work order with a status other than **Closed**. Go to the **Features** tab of this work order. The feature "Example1" is not displayed but you can click  to add this optional field.

The script fulfills its function correctly.

## Second example

This section deals with a hypothetical problem that can be solved using a Basic script. We recommend trying this problem yourself before consulting the described solution.

## Description of the problem

By default, we want the **Field1** field (SQL name: Field1) in the detail of an employee show the name and first name of the employee if both of these exist, or just the name if the first name is missing.

## Step 1: Analyze and define the algorithm

The algorithm must fulfill the following tasks:


- Display by default the name and the first name of the employee in **Field1** (SQL name: Field1) in the employee detail if both the name and first name exist.
- Display by default the name alone of the employee in **Field1** (SQL name: Field1) in the employee detail if the first name does not exist.

Our algorithm is therefore as follows:

```
If the first name of the employee does not exist Then
The default value of "Field1" is the name of the employee
Else
The default value of "Field1" is "Last name," "First name"
```

It is therefore the values of the **Name** (SQL name: Name) and **First** (SQL name: FirstName) fields in the table of employees which determine the default value of **Field1** (SQL name: Field1) the employee detail. Only these two fields appear in our algorithm.

Right-click **Field1** (SQL name: Field1) in the employee detail and select **Configure object**.

Click the magnifier button  next to the **Default** field to edit the Basic script.

## Step 2: Draw up the Basic script

You now need to write the algorithm from step 1 in Basic.

```
If [FirstName]= "" Then
RetVal = [Name]
Else
RetVal = [Name]+", "+[FirstName]
End If
```

Click **OK** to confirm your script.

## Step 3: Test the Basic script

This step allows you to make sure that the script functions correctly.

1. Open the table of employees and departments by selecting the **Organization management/ Organization/ Employees and departments** link on the navigation bar, and create a new employee.
2. Populate the **Name** (SQL name: Name) and **First** (SQL name: FirstName) fields, then click **Create** to confirm. Asset Manager displays the name and first name of the employee in **Field1** (SQL name: Field1).

The script fulfills its function correctly.

# Chapter 11: Calendars

Use the **Organization management/ Operations/ Calendar of business days** navigation menu item to access the list of calendars.

This chapter includes:

|                                                             |     |
|-------------------------------------------------------------|-----|
| Overview of calendars .....                                 | 253 |
| Impact of calendars on certain areas of functionality ..... | 254 |
| Methodology used to create a calendar .....                 | 255 |
| Description of how to create a calendar .....               | 255 |

## Overview of calendars

This section includes:

|                           |     |
|---------------------------|-----|
| The calendar detail ..... | 253 |
| Using calendars .....     | 254 |

## The calendar detail

The detail of a calendar shows:

- General information that enables you to uniquely identify the calendar:
  - The **Name** (SQL name: Name) of the calendar.
  - The **Time zone** (SQL name: TimeZone) to which the calendar is attached.
- The description of usual business hours in the **Timetables** tab.
- The list of exceptions to this timetable in the **Exceptions** tab.
- A preview of business hours for a given period that takes into account the rules defined in the **Timetable** and **Exceptions** tabs.

## Using calendars

Calendars are associated with:

- Helpdesk tickets via escalation schemes.
- Workflow activities.

They allow you to set the time when alarms defined in escalation schemes or in workflow activities are to be triggered. The **Time limit** defined in these alarms is, in effect, specified in business hours.

Example:

1. You create a ticket on Thursday September 25, 2000 at 17:55.
2. The ticket is associated with an escalation scheme that contains an alarm to be triggered if the ticket is not assigned within the 15 minutes following its opening.
3. The escalation scheme is associated with a calendar that specifies that the periods on Thursday September 25 and Friday September 26 from 09:00 to 18:00 are workable.
4. The alarm will be triggered on Friday September 26, 2000 at 09:10 if the ticket is not assigned at this moment.

If no calendars are attached to an escalation scheme, all hours are considered as being business hours.

**Caution:** If you modify a calendar in the database, modifications are only taken into account at the level of those fields linked to the calendar when you exit and relaunch Asset Manager.

For example, if a ticket is associated with an escalation scheme linked to a modified calendar, the **Expected resol.** (SQL name: dtResolLimit) date in the **Tracking** tab of the ticket detail is updated the next time Asset Manager is relaunched.

## Impact of calendars on certain areas of functionality

Calendars have an impact on certain areas of functionality of Asset Manager. Modifying a calendar brings about changes both directly and indirectly to certain records in the database. Calendars are involved in:

- Escalation schemes. They set the time at which alarms are triggered.
- Activities link to the processing of helpdesk tickets.
- The resolution time of helpdesk tickets (expressed in business hours).
- The resolution dates of helpdesk tickets.
- The suspension time of helpdesk tickets.
- The execution times of workflow tasks.
- Alarms associated with workflow activities.

## Methodology used to create a calendar

Use the following steps to create a calendar:

1. Start by identifying the calendar by giving it a **Name** (SQL name: Name).
2. If necessary, associate the calendar with a time zone by populating the **Time zone** field (SQL name: TimeZone).
3. Define the daily business hours in the **Timetables** tab of the calendar detail.
4. Next, define any exceptions to these business hours in the **Exceptions** tab of the calendar detail.
5. Finally, you can verify the functioning of the calendar via the **Preview** tab.

## Description of how to create a calendar

A calendar is created step by step:

|                                     |     |
|-------------------------------------|-----|
| Entering general information .....  | 256 |
| Populating the Timetable tab .....  | 256 |
| Populating the Exceptions tab ..... | 257 |
| Previewing the calendar .....       | 260 |

## Entering general information

Before entering the actual business hours and exceptions, you must identify the calendar by populating the **Name** field (SQL name: Name) in the detail screen.

You can also associate the calendar with a time zone by populating the **Time zone** field (SQL name: TimeZone).

This facilitates processing tickets that involve assets or users situated in different time zones. Time limits involved in processing such tickets are based on time zone information.

## Populating the Timetable tab

The **Timetables** tab in a calendar detail defines the weekly timetables associated with this calendar. The periods of business hours defined in this tab define the weekly timetables associated with this calendar. These periods describe the general rule. Public holidays and the like constitute exceptions and are defined in the **Exceptions** tab.

### Calendar - Timetables tab

| Day       | Time Range                       |
|-----------|----------------------------------|
| Monday    | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Tuesday   | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Wednesday | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Thursday  | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Friday    | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Saturday  |                                  |
| Sunday    |                                  |

For each day of the week, you can define one or more timetable periods representing business hours. You can define these using two methods:

- Graphically by using the graduated slider controls representing each day of the week.
  - a. Click the control at the start of the timetable period.
  - b. Extend the selection by dragging the mouse to the end of the timetable period. Asset Manager automatically populates the text field to the right of the graduated control.
  - c. Repeat as necessary.
- "Manually", using the text field. Use the following syntax for this field:



<First hour of the business period>-<Last hour of the business period>;< First hour of the business period >-< Last hour of the business period >;...

The following format is used for time values:

<hh:mm[AM|PM]>

If the optional [AM|PM] parameter is not defined, Asset Manager considers that the 24 hour format is used.

Asset Manager automatically populates the graduated slider control situated to the left of the text field.

**Note:** Using the graphical control only gives you a precision to the nearest half-hour. Using the text control is accurate to the nearest minute.

## Populating the Exceptions tab

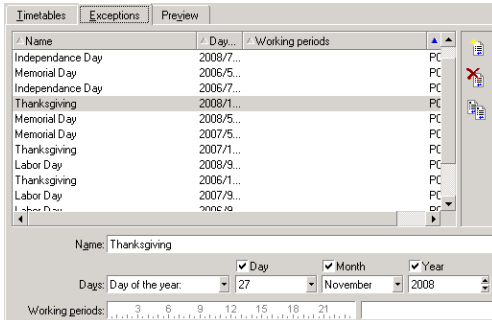
The **Exceptions** tab in the calendar detail defines exceptions to the weekly business periods defined in the **Timetables** tab.

### Methodology




To create an exception:

1. Give a "Name" to the exception.
2. Define the field of application of the exception by populating the **Days** field. Exceptions can be defined according to a given day, month or year.
3. It is also possible to define a business period within an exception using the **Working periods** field. This field allows you define more precise exceptions such as: "On the last Friday of each month, the team works from 08:30 to 10:30 and from 17:30 to 18".

## The Exceptions tab details



This tab is divided up into two parts.

- The first part gives you a list of exceptions and allows you to create, duplicate, destroy, modify, and cancel modifications using the buttons on the toolbar:
  -  : Click this button to create a new exception.
  -  : Click this button to delete an exception.
  -  : Click this button to duplicate an exception.

**Note:** The **Ranking** column allows you to sort exceptions by priority: It determines which exception takes priority in case of conflict. Asset Manager automatically assigns a ranking (from "P00" to "P15") to an exception. The smaller the number, the higher the priority of the exception. Thus an exception ranked "P06" takes priority over an exception ranked "P10".

- The second part gives you the details of the exception.  
The values taken by the **Days** field define the context of application of the exception:

| Value of the Days field | Context of application of the exception                                                                                                                                                                                      |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "Daily"                 | The exception applies to all days of the year without exception.                                                                                                                                                             |
| "Day of the year:"      | The exception applies to given day or selection of days, defined using the <b>Day</b> , <b>Month</b> and <b>Year</b> check boxes.                                                                                            |
| "The first"             | The rule applies to the weekday defined via the <b>Day</b> check box, and for the month (s) and year(s) defined via the <b>Month</b> and <b>Year</b> check boxes.<br><br><b>Example</b><br>"The first" Friday of each month. |

| Value of the Days field | Context of application of the exception                                                                                                                                                                                                                      |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "The second"            | <p>The rule applies to the weekday defined via the <b>Day</b> check box, and for the month (s) and year(s) defined via the <b>Month</b> and <b>year</b> check boxes.</p> <p><b>Example</b></p> <p>"The second" Monday of the month of September.</p>         |
| "The next to last"      | <p>The rule applies to the weekday defined via the <b>Day</b> check box, and for the month (s) and year(s) defined via the <b>Month</b> and <b>year</b> check boxes.</p> <p><b>Example</b></p> <p>"The next to last" Wednesday of the month of November.</p> |
| "The last"              | <p>The rule applies to the weekday defined via the <b>Day</b> check box, and for the month (s) and year(s) defined via the <b>Month</b> and <b>Year</b> check boxes.</p> <p><b>Example</b></p> <p>"The last" Tuesday of each month throughout 2000.</p>      |

## Example

Employees at Taltek have the following days off:

### **Rule No.1: The first Friday of each month is a day off.**

1. Click **New** to begin the creation of the exception.
2. The exception applies the first Friday of every month for all years. The **Month** and **Year** check boxes are therefore unchecked, meaning that the exception is independent of the month and the year. The **Day** box is checked, since the exception only applied to Fridays.
3. To end: Set the **Days** field to: "The first".

### **Rule No.2: During August, employees of Taltek only work the morning from 08:30 to 12:30.**

1. Click **New**.
2. This exception is only dependent on the month (applies to the month of August only). The **Day** and **Year** boxes are therefore unchecked, the **Month** box is checked and the corresponding value set to "August".
3. Since the exception only applies to a selection of days (all days throughout the month of August), the **Days** field just needs to be set to "Day of the year".

4. During this period, employees work from 08:30 to 12:30. To finish entering the exception you just need to select the period from 08:30 to 12:30 in the **Working periods** field.

## Previewing the calendar

The **Preview** tab allows you to apply the rules defined in the **Timetables** and **Exceptions** tabs to a period selected using the **Start date** and **End date** fields in the **Test** zone.

### Calendar - Preview tab

| Start date            | End date              | Time elapsed |
|-----------------------|-----------------------|--------------|
| 12/16/2010 1:27:33 PM | 12/16/2010 1:27:33 PM | 0h           |

| Calendar associated with the start date | 0 | 4 | 8 | 12 | 16 |                                  |
|-----------------------------------------|---|---|---|----|----|----------------------------------|
| Monday, December 13, 2010               |   |   |   |    |    | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Tuesday, December 14, 2010              |   |   |   |    |    | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Wednesday, December 15, 2010            |   |   |   |    |    | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Thursday, December 16, 2010             |   |   |   |    |    | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Friday, December 17, 2010               |   |   |   |    |    | 9:00 AM-12:00 PM;1:00 PM-6:00 PM |
| Saturday, December 18, 2010             |   |   |   |    |    |                                  |
| Sunday, December 19, 2010               |   |   |   |    |    |                                  |

- The **Calendar associated with the start date** frame by default gives you a preview of the business hours during the week containing the selected "Start date".
- The **Time elapsed** field gives the total number of business hours during the selected period.

**Note:** You can enter a duration in the **Time elapsed** field to force the recalculation of the **End date** field according to the given **Start date**.

# Chapter 12: Time zones

This chapter explains how to use time zones.

**Note:** This area of functionality is only available for certain Asset Manager license agreements.

This chapter includes:

|                                                                      |     |
|----------------------------------------------------------------------|-----|
| Why manage time zones? .....                                         | 261 |
| Implementing time zones .....                                        | 262 |
| Creating time zones .....                                            | 262 |
| Managing a time zone .....                                           | 263 |
| Managing time zones in Asset Manager Automated Process Manager ..... | 269 |
| Consequences for various operations .....                            | 270 |

## Why manage time zones?

Since client machines and the database server can be separated geographically, Asset Manager manages time zones and time differences in relation to Greenwich Mean Time (GMT). Asset Manager respects the following rules:

- All "Date and time" type fields are displayed on the client machine respecting the time zone of the client machine.
- All "Date and time" type fields are stored on the server in reference to a defined time zone.
- All calculations involving dates and times take time zone differences into account.

### Example

Let's take the example of a server situated in New York that has data indexed according the Paris (France) time zone and two client machines situated in London and in Paris. Let's start by defining the time zone of each of these client machines according to Greenwich Mean Time:

- Time zone of server = GMT-5
- Time zone of Paris client = GMT+1

- Time zone of London client = GMT
- Time zone of data = GMT+1

All "Date and time" type values are thus stored on the server in GMT+1 format and are displayed on the Paris client as GMT+1 and on the London client as GMT. For example, when taking a work order on the London client machine, if the resolution deadline of the work order is set to May 15, 2000 at 17:30, this is shown as follows on the other machines:

- On the server: May 15, 2000 at 12:30.
- On the Paris client: May 15, 2000 at 18:30.
- On the London client; May 15, 2000 at 17:30.

## Implementing time zones

In order for time zones to be handled correctly under Asset Manager you need to follow the following steps:

1. Define the time zones when creating the database under Asset Manager Application Designer using the **Use time zones** option.
2. Create the time zones (by importing the information relative to time zones, for example).
3. Define the time zone of your machine using the **Tools/ Change the time zone** menu item.

**Note:** This option only appears when the time zone was defined when creating the database.


4. Define the calendars in accordance with the time zones.

## Creating time zones

The time-zone functionality in Asset Manager, unlike in Windows, handles modifications to daylight-saving rules over the years. This enables you to display local times in the past with greater accuracy. Asset Manager's use of time zone information allows you to:

- Display local dates and times taking into account daylight saving changes.
- Put yourself in the place of another location.

In order to avoid having to define time zones manually, Asset Manager ships with a description file containing the principal time zones. This file can be imported using the following procedure:

1. Select the **File/ Import** menu item in the Windows client. Asset Manager opens the import selection window.
2. Select "Execute a script" by clicking . Asset Manager opens the database update screen. Click  to select the script to execute, in this case **tz.scr** in the **datasys** sub-folder of the Asset Manager installation folder.
3. Click **Import**. Asset Manager performs the import according to the script.

## Managing a time zone

In this section, we will look at the **Daylight time** field (SQL name: memDaylightInfo) in more detail.

This section includes:

|                                             |     |
|---------------------------------------------|-----|
| Format of Daylight time field .....         | 263 |
| Values of the <Year> argument .....         | 264 |
| Values of the <DaylightInfo> argument ..... | 264 |
| Example .....                               | 267 |

## Format of Daylight time field

The **Daylight time** field (SQL name: memDaylightInfo) is structured as follows (in one single line):

```
<Year>=<DaylightInfo>|<Year>
=<DaylightInfo>|<Year>=<DaylightInfo>|...
```

In the rest of this section, the following conventions will be used:

- <Year>=<DaylightInfo> together is named the "parameter"
- <Year> and <DaylightInfo> separately are named "arguments"

The table below gives you an overview of daylight savings changes according to the values of the <Year> and <DaylightInfo> arguments.

|                               | The <DaylightInfo> argument is empty            | The <DaylightInfo> argument has a value                                                             |
|-------------------------------|-------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| The <Year> argument is empty. | There is no change for daylight savings for the | Daylight savings information is valid for all years, excepting those defined by parameters having a |

|                                 | The <DaylightInfo> argument is empty | The <DaylightInfo> argument has a value                                                                                                                   |
|---------------------------------|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| ("<Year>=" does not appear)     | entirety of this time zone.          | <Year> argument.                                                                                                                                          |
| The <Year> argument has a value | Not applicable                       | Daylight savings information for this time zone is valid for every year from the year specified by the <Year> argument up until the next <Year> argument. |

## Values of the <Year> argument

The <Year> argument that specifies the year from which the daylight saving information defined in the <DaylightInfo> is applicable, can take any four figure year value (such as 1990, 1997, 1998, 2012).

## Values of the <DaylightInfo> argument

The full format of a <DaylightInfo> argument is as follows (in one single line):

```
<StdShift>, <DltShift>, <SDay>
, <SMonth>, <SDayPos>, <SHour>
, <DDay>, <DMonth>, <DDayPos>, <DHour>
```

This argument is made up of several sub-arguments as shown below:

| Sub-argument | Description                                                                                                                                                                                                                                                                                          | Possible values                                                                                                                         |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <StdShift>   | Expressed in minutes, this describes the time difference between standard time within a time zone and the time of the time zone concerned.<br><br>For example, for Paris (GMT+1 time zone), if <StdShift> is set to 30 (minutes), standard time within this time zone is GMT+1h30min and not GMT+1h. | By default, this sub-argument is null, but it can be set to any numeric value. The user must verify the coherency of this sub-argument. |
| <DltShift>   | Expressed in minutes, this                                                                                                                                                                                                                                                                           | By default, this sub-argument is set to 60 (which                                                                                       |



| Sub-argument | Description                                                                                         | Possible values                                                                                                                                                                                                                                      |
|--------------|-----------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | describes the time difference between daylight saving time and the time of the time zone concerned. | corresponds to 1 hour's time difference between daylight saving time and the reference time "GMT +") but it can be set to any numeric value. The user must verify the coherency of this sub-argument.                                                |
| <SDay>       | Day of change from daylight saving time to standard time.                                           | "Monday"<br>"Tuesday"<br>"Wednesday"<br>"Thursday"<br>"Friday"<br>"Saturday"<br>"Sunday"<br><br>Empty (in this case, you need to set <SDayPos> to a value between 1 and 31 to identify the day of change from standard time to daylight saving time) |
| <SMonth>     | Month of change from daylight saving time to standard time.                                         | "January"<br>"February"<br>"March"<br>"April"<br>etc.<br>"November"<br>"December"                                                                                                                                                                    |
| <SDayPos>    | Position in the month of the day of change from daylight saving time to standard time.              | "First"<br>"Second"<br>"Third"<br>"Fourth"<br>"Last"<br>"Penultimate" (before-last)<br>A value between 1 and 31 when <SDay> is empty                                                                                                                 |
| <SHour>      | Time of change from daylight saving time to standard time (expressed in daylight saving time).      | Any value expressed in 24 hour format (HH:MM:SS).                                                                                                                                                                                                    |

| <b>Sub-argument</b> | <b>Description</b>                                                                      | <b>Possible values</b>                                                                                                                                                                                                                                 |
|---------------------|-----------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <DDay>              | Day of change from standard time to daylight saving time.                               | "Monday"<br>"Tuesday"<br>"Wednesday"<br>"Thursday"<br>"Friday"<br>"Saturday"<br>"Sunday"<br><br>Empty (in this case, you need to set <DDayPos> to a value between 1 and 31 to identify the date of change from daylight saving time to standard time.) |
| <DMonth>            | Month of change from standard time to daylight saving time.                             | "January"<br>"February"<br>"March"<br>"April"<br>etc.<br>"November"<br>"December"                                                                                                                                                                      |
| <DDayPos>           | Position in the month of the day of change from standard time to daylight saving time.  | "First"<br>"Second"<br>"Third"<br>"Fourth"<br>"Last"<br>"Penultimate" (before last)<br>A value between 1 and 31 when <DDay> is empty                                                                                                                   |
| <DHour>             | Time of change from standard time to daylight saving time (expressed in standard time). | Any value expressed in 24 hour format (HH:MM:SS).                                                                                                                                                                                                      |

## Example

As an example, let's take the information for daylight saving time for "(GMT+01:00) Paris, Madrid, Amsterdam" time zone.

```
2000=0,60,Sunday,October,last,03:00:00,Sunday,March,last,02:00:00|
0,60,Sunday,September,last,03:00:00,Sunday,March,last,02:00:00
```

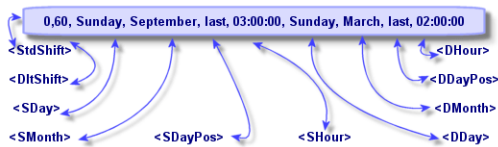
Let's now identify the parameters used.

### First parameter



- <Year> = 2000 means that the following <DaylightInfo> parameters are applicable from 2000 onwards.
- <StdShift> = 0 means that there is no difference between the time zone and standard time within this time zone.
- <DltShift> = 60 means that the time difference between standard time and daylight saving time is 60 minutes, that is, 1 hour. Daylight saving time is therefore equal to the time of the time zone plus one hour.
- <SDay> = Sunday means that the change to standard time takes place on a Sunday.
- <SMonth> = October means that the change to standard time takes place during the month of October.
- <SDayPos> = Last specifies the position of the day in the month. Here the change to standard time takes place on the last Sunday of the month of October.
- <SHour> = 03:00:00 means that the change to standard time takes place at 03:00.
- <DDay> = Sunday means that the change to daylight time takes place on a Sunday.
- <DMonth> = March means that the change to daylight time takes place in the month of March.
- <DDayPos> = Last specifies the position of the day within the month. In this case, the change to daylight time takes place on the last Sunday of the month of March.
- <DHour> = 02:00:00 means that the change to daylight takes place at 02:00.

## Second parameter



- Since there is no <Year> argument, this parameter specifies that it is only valid for those years not described in the previous parameter.
- <StdShift> = 0 means that there is no difference between the time zone and standard time within this time zone. Winter time is thus equal to the time zone time.
- <DltShift> = 60 means that the time difference between standard time and daylight saving time is 60 minutes, that is, 1 hour. Daylight saving time is therefore equal to the time of the time zone plus one hour.
- <SDay> = Sunday means that the change to standard time takes place on a Sunday.
- <SMonth> = September means that the change to standard time takes place during the month of September.
- <SDayPos> = Last specifies the position of the day in the month. Here the change to standard time takes place on the last Sunday of the month of September.
- <SHour> = 03:00:00 means that the change to standard time takes place at 03:00.
- <DDay> = Sunday means that the change to daylight time takes place on a Sunday.
- <DMonth> = March means that the change to daylight time takes place in the month of March.
- <DDayPos> = Last specifies the position of the day within the month. In this case, the change to daylight time takes place on the last Sunday of the month of March.
- <DHour> = 02:00:00 means that the change to daylight takes place at 02:00.

As a result:

**Note:** From 2000 onwards, the change to standard time is made on the last Sunday of October at 03:00:00 (clocks go backward to 02:00:00) and the change to daylight saving time takes place on the last Sunday of March at 02:00:00 (clocks go forward to 03:00:00).

For all years before 2000, the change to standard time is made on the last Sunday of September at 03:00:00 and the change to daylight saving time takes place on the last Sunday of March at 02:00:00.

# Managing time zones in Asset Manager Automated Process Manager

Asset Manager Automated Process Manager enables you to configure tests concerning time zones. Select the **Tools/ Configure modules** menu item.

This section includes:

|                             |     |
|-----------------------------|-----|
| Tests to perform .....      | 269 |
| Frequency of the test ..... | 270 |

## Tests to perform

In the **General** tab of the configuration screen, you configure the type of test to be performed:

- Verify time zone of database server.
- Verify local time compared to that of the server.

These two tests compare the time of the database server with that of the machine on which Asset Manager Automated Process Manager is installed. The time difference is expressed as  $[(n * 30\text{minutes}) + m]$  where  $m$  is between  $-15$  minutes and  $+ 15$  minutes.

### In both cases

If the time difference exceeds 5 minutes, Asset Manager Automated Process Manager offers to update the local time on the machine on which it is installed.

If you refuse this update (for example, if you think that it is the time of the server that requires changing), the connection is refused. You can connect again once the difference between the two times no longer exceeds 5 minutes (resulting from either a modification to the time of the database server and/or of the machine on which Asset Manager Automated Process Manager, is installed).

### Specific aspects of the Verify time zone of database server option

If necessary, the information on the time zone of the server in the table of options of Asset Manager is updated (if the number  $(n * 30 \text{ minutes})$  does not correspond to the time zone of the server).

**Note:** To do this, the machine on which Asset Manager Automated Process Manager is running must have the correct time and have the correct information on daylight saving changes.

Specific aspects of the Verify local time compared to that of the server option

The time zone of the server, necessary for internal operations of Asset Manager, is recovered.

## Frequency of the test

The test is performed:

1. First, when Asset Manager Automated Process Manager connects to the database.
2. Then regularly, according to the schedule you specify in Asset Manager Automated Process Manager (**Tools/Configure modules**).

## Consequences for various operations

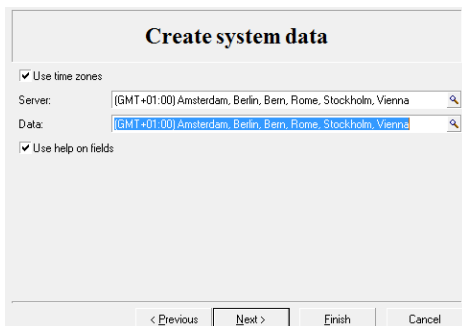
The way time zones are managed impacts a certain number of operations:

This section includes:

|                                        |     |
|----------------------------------------|-----|
| Creating the database .....            | 270 |
| Connecting to a database .....         | 271 |
| Import and Export .....                | 273 |
| Calendars and escalation schemes ..... | 273 |

## Creating the database

When creating a database, Asset Manager allows you to define time zone options. Select the **Action/ Create database** menu item in Asset Manager Application Designer. The **Create system data** frame contains the options concerning time zones.



The **Use time zones** check box determines whether time zones are taken into account when creating the database.

- If the box is checked, time zones are used when creating the database.
- If this check box is cleared, time zones are ignored when creating the database.

The **Server** and **Data** fields determine the effective time zone of the server when the database was created and the time zone to be used for the stored data.

**Note:** This option is only accessible when creating a database. It allows you to define the time reference of the server and of the data. If you change these values, the "Date time" values already contained in the database do not make sense.

## Connecting to a database

When connecting to a database, Asset Manager searches in **am.ini** for the "LocalTimeZone" entry that defines the time zone of the client machine.

Location of this file: See **Asset Manager - Installation and upgrade** guide, chapter **.ini and .cfg files**.

If this information cannot be found, Asset Manager uses the system time zone (defined in Windows).

Asset Manager next searches the database for the time zone corresponding to the "LocalTimeZone" entry in **am.ini**, or the time zone defined in Windows.

The table below summarizes the different possible situations:

| "LocalTimeZone" exists in am.ini? | Corresponding time zone found in table of time zones? | Information saved in am.ini file for the "LocalTimeZone" entry |
|-----------------------------------|-------------------------------------------------------|----------------------------------------------------------------|
| Yes                               | Yes                                                   | Database time zone                                             |
|                                   | No                                                    | Unchanged                                                      |
| No                                | Yes                                                   | Database time zone                                             |
|                                   | No                                                    | System time zone                                               |

## Adjusting time on client machine

On connecting to a remote database, Asset Manager verifies the validity of the time given by the clock at client level with reference to the server clock.

The clock differential is a difference in synchronization and is not to be confused with the time difference, which is a transposition in relation to a difference in time zones. Asset Manager calculates the time zone of the client clock and determines the clock differential between the two machines. This calculation is made as follows:

Differential = Modulo((Difference in minutes between the times of the two machines in question)/30)

**Note:** The modulus is the remainder of a division.

For example, for the following machines:

- Machine A is in GMT and is showing 18:02
- Machine B is in GMT+1 and is showing 18:19 (that is, 17:19 for machine A, 43 minutes ' difference with machine A))

Differential = Modulo (43/30)= 13 minutes

If this difference exceeds five minutes (fixed value), Asset Manager proposes adjusting clock time at the client level.

Connection fails if the user refuses.

Asset Manager performs this check periodically and when time is changed on the client machine. By default, this check is made every 60 minutes, but this can be configured by modifying the **g\_ ITimeZoneCheckInMns** option in **am.ini**, in the [option] section.

Location of this file: see **Asset Manager - Installation and upgrade** guide, chapter **.ini and .cfg files**.



```
[option]
g_lTimeZoneCheckInMns = 30
```

The frequency of the clock differential check is set to 30 minutes.

This frequency can also be configured via the **Verify database server time zone** option in Asset Manager.

**Note:** This verification only functions if the database concerned has been created with time zone taken into account.

## Import and Export

For these two functions, the conversion is made assuming that all "Date and time" fields correspond to the time zone of the machine carrying out the import or export.

## Calendars and escalation schemes

Calendars can be associated with time zones.

As a result:

- Time values expressed in the calendar are in local time, in relation to the selected time zone.
- It is up to the user to select a calendar that is compatible with the escalation schemes.

# Chapter 13: Calculated fields

You access the screen to create calculated field via the **Administration/ System/ Calculated fields** menu.

This chapter includes:

|                                        |     |
|----------------------------------------|-----|
| Definition of a calculated field ..... | 274 |
| Usefulness of calculated fields .....  | 274 |
| Creating a calculated field .....      | 275 |
| Using calculated fields .....          | 280 |

## Definition of a calculated field

A calculated field is a field whose value is calculated according to the value of other fields and variables, using a user-defined formula. There are three types of calculated field:

- AQL
- Basic
- Calculated field

Each of these types uses a different language for the calculation formulas and has an effect on the possibilities and constraints linked to using the field. For example, only "AQL" type calculated fields can be used in filters.

**Note:** Calculated fields are virtual fields that are read-only (the formula alone is stored in the database). You can define as many calculated fields as you like and assign user rights to them.

## Usefulness of calculated fields

Calculated fields make it possible to define additional information and to calculate synthetic information concerning records of a table in the Asset Manager database. In this way, apart from a few differences, they resemble "classic" database fields:

- Unlike "classic" fields, the value of a calculated field is not stored in the Asset Manager database.
- The value of a calculated field is not populated by a user but given by a formula.
- You cannot associate a calculated field with a single record of a given field. Just like all other "classic" fields in the database, a calculated field is associated with all the records of a table and has a value (can be null) for each record in this table.
- Calculated fields do not appear in the detail screen of a record. They can only be displayed in a list.
- Calculated fields can only be used in the calculation of the default values of standard fields if their type is **Calculated string** or **Basic script**.

## Creating a calculated field

Before creating such a field, it is useful to be acquainted with the specific details inherent to each type of calculated field.

This section includes information on the following topics:

|                           |     |
|---------------------------|-----|
| Introduction .....        | 275 |
| Methods of creation ..... | 277 |

## Introduction

Each type of calculated field has different properties that determine how it is used.

The following table resumes the main differences between the three types:

### Calculated fields types

| Type of field | Properties of fields of this type |               |                        | Field calculated by | Characteristics of the language used by the calculation formula |                                                    |
|---------------|-----------------------------------|---------------|------------------------|---------------------|-----------------------------------------------------------------|----------------------------------------------------|
|               | Can be displayed                  | Can be sorted | Can be used in filters |                     | Advantages                                                      | Disadvantages                                      |
| AQL           | Yes                               | Yes           | Yes                    | The database server | Powerful<br>Integrated                                          | Limited language.<br>Fields of this type cannot be |

**Calculated fields types, continued**

| Type of field      | Properties of fields of this type |     |    | Field calculated by | Characteristics of the language used by the calculation formula |                                                                                            |
|--------------------|-----------------------------------|-----|----|---------------------|-----------------------------------------------------------------|--------------------------------------------------------------------------------------------|
|                    |                                   |     |    |                     | editor                                                          | used in default values.                                                                    |
| Calculated strings | Yes                               | Yes | No | The client          | Simple                                                          | Not very powerful (simple concatenation of strings and values of fields and strings only). |
| Basic              | Yes                               | No  | No | The client          | Many possibilities<br>Flexible                                  | Fields of this type can be displayed only.                                                 |

**Note:** This table clearly shows that "AQL" type fields have much wider range of application than the two other types of calculated fields.

An AQL query can make use of all three properties (can be displayed, sorted, used in filters):

| Property               | Corresponding AQL arguments                               |
|------------------------|-----------------------------------------------------------|
| Can be displayed       | SELECT clause                                             |
| Can be sorted          | SELECT<br>ORDER BY<br>GROUP BY clauses                    |
| Can be used in filters | SELECT<br>ORDER BY<br>GROUP BY<br>WHERE<br>HAVING clauses |

For further information on AQL queries, refer to chapter **AQL queries** of this guide.

**Calculation by the server / client machine**

In the case of an "AQL" type field, the database server performs the necessary calculations and sends the result to the client machine. For this reason, there is no impact on the speed of the client machine

and network traffic is reduced. On the other hand, the SQL queries sent to the database are more complex.

## Methods of creation

This section describes in detail the method used to create a calculated field.

### Analyze your needs

Choose a type of field based on the two following criteria:

- The properties of the type of field: Can be displayed, sorted, used in filters or default values.
- The "cost" of this solution; in terms of complexity of the formula used compared to the possibilities of utilization. The three types of calculated field can be classified as follows (in terms of increasing cost):
  - Calculated string
  - AQL
  - Basic

**Note:** Whenever possible, we recommend using the least "costly" solution.

For example:

- If the field is for informational purposes only, a Basic type calculated field will suffice.
- If you want to be able to sort records according to the value of the field, "AQL" or "Calculated string" must be used.
- If you want to be able to filter records according to the value of the field, "AQL" must be used.

Once you have defined your needs, you can move on to the next phase.

### Open the creation screen

Select the **Administration/ System/ Calculated fields** menu item. Asset Manager displays the calculated field creation screen:

The screenshot shows a configuration window for a calculated field named 'Basic'. The 'Title' field is 'Basic', the 'SQL name' is 'Basic', and the 'Description' is 'Basic'. The 'Table' is set to 'Assets [amAsset]', the 'Field type' is 'Basic script', and the 'Result type' is 'Text'. There is a checkbox for 'Only for programming'. Below these fields is a 'BASIC syntax' section with a code editor containing the following script:

```

If Left([AssetTag], 2)='PC' Then
RetVal='PC'
Else
RetVal='Other'
End If

```

## Identify the calculated field

First, populate the upper part of this screen to uniquely identify the calculated field:

- The **Title** field (SQL name: Label) contains the label of the calculated field, used for column headers in lists.
- The **SQL name** field (SQL name: SQLName) contains the SQL name of the calculated field. This name, prefixed by the letters "cf\_", is used, for example, when referring to this field in Basic scripts, queries or filters.

**Note:** You must not modify the SQL name of a field once it has been created. Any references to this field using the previous SQL name will no longer be valid.


- The **Description** field (SQL name: Description) contains a short description of the field, used in the lists showing the fields (in filters or the list configuration screen, for example).

## Define the context of utilization of the field

The **Table** (SQL name: TableName) and **Field type** (SQL name: seType) fields allow you to define the context of utilization of a calculated field:

- The **Table** field (SQL name: TableName) allows you to associate a calculated field with a table. The field will only be available for this table.
- The **Field type** field (SQL name: seType) allows you to specify the type of the calculated field. According to this type, the properties of the field (can be displayed, sorted or used in filters) will differ.
- The **Result type** field allows you to specify the resulting type of the calculated field. This type is used for formatting and display purposes. A calculated field whose result type is a date is thus displayed in the same way as all other "Date" type fields in the database.

## Enter the calculation formula for the field

Now you just have to write the field calculation formula. You can either enter this directly in the text field in the lower part of the screen (note that the label of this field changes according to the attributed type), or click the magnifier  or press "F4" to access the corresponding editor screen.

**Note:** The language used differs according to the type of field used.


For further information on the languages available for writing a calculation formula, Refer to the following documentation:

- Chapter **Scripts** of this guide. The function used is **RetVal()**.
- Chapter **AQL queries** in this guide for the AQL language.
- **Administration** manual, chapter **Standard database description files**, section **Description of the tables**, sub-section **Table description strings** for calculated strings.

## Define the user rights for the calculated field

Select the **Administration/ Rights/ User rights** navigation menu. Asset Manager displays the screen for creating user rights.

**Note:** Calculated fields are accessible for status access only.

1. Enter a brief description for the user right in the **Description** (SQL name: Description) field (SQL name: Description) and, optionally, a comment in the **Comment** field (SQL name: Comment).
2. If you are using the Windows client, expand the tree structure of the table associated with the calculated field. The branch identified by the  icon includes a full list of the calculated fields for the table in question.  
If you are using the Web client, select the table and then select the calculated field from the category list of the table components.
3. Then select the field for which you want to edit a user right. The **Read** check box in the **Fields, links and features** zone allows you to define the read rights for this field. When this box is checked, only profiles with this user right can view the calculated field. If the box is not checked, then all users will have (read-only) access to this field.

## Using calculated fields

How a calculated field can be used depends on its type. You need to make sure that the type is compatible with its intended utilization. In lists showing fields (creating a filter, configuring a list, and so on), Asset Manager helps you by only showing those fields that can be used.

This section includes:

|                                                               |     |
|---------------------------------------------------------------|-----|
| Using a calculated field in the configuration of a list ..... | 280 |
| Filtering the records of a table .....                        | 280 |
| Referencing a calculated field .....                          | 280 |

## Using a calculated field in the configuration of a list

You can display the value of a field calculated for all records in a table using the **Configure list** command in the shortcut menu.

**Caution:** If the DBMS is Microsoft SQL Server and you have enabled one of the **Applied to get query generating** and **Applied to select query generating** database options, you must use the SQL Server native client as your ODBC driver.

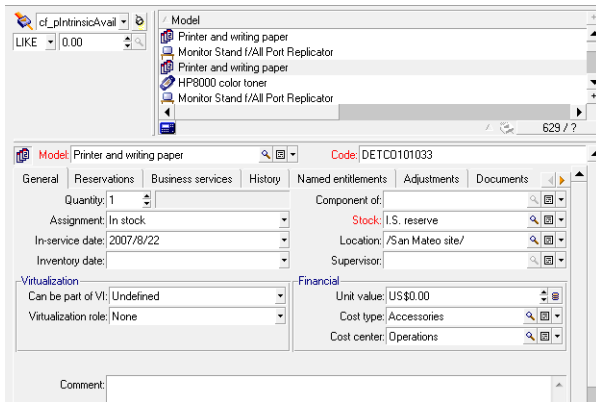
## Filtering the records of a table

Asset Manager can filter the records of a table according to the value of an "AQL" type calculated field. To do this, select the **Simple filter** command from the shortcut menu and go to the **Calculated fields** branch. Asset Manager only shows the "AQL" type fields.

## Referencing a calculated field

The SQL name, prefixed by the letters "cf\_" is used to reference calculated fields. The screen copy below illustrates the use of the SQL name of a calculated field in a filter:





Additionally, calculated fields can be used by the different modules or functions of Asset Manager:

- Asset Manager Web
- Asset Manager API
- Reports

For further information on these modules or areas of functionality, Refer to the corresponding documentation.

# Chapter 14: Wizards

Asset Manager lets you create your own wizards and adapt existing wizard for your own use. Wizards are stored as text fields (**Wizard script** field (SQL name: WizardScript), **Wizard** tab of **Assistant** type action detail). Creating a wizard consists of entering its code in this field directly or using the graphical editor. Doing this requires familiarity with the structure of wizards and the scripting language used to describe this structure.

**Caution:** If you modify script or wizard type actions, make sure you re-tag the Web services if you have deployed Asset Manager Web.

See **Tailoring** guide, chapter **Customizing the database**, section **Development best practices/ Tag the Web services**.

This chapter includes:

|                                                   |     |
|---------------------------------------------------|-----|
| Overview .....                                    | 282 |
| Sequencing wizards .....                          | 288 |
| Basic functions .....                             | 289 |
| Properties of a node .....                        | 291 |
| Types of nodes .....                              | 296 |
| Types of controls and associated properties ..... | 310 |
| Using the graphical editor .....                  | 338 |
| Example of creating a wizard .....                | 342 |
| Wizard programming case study .....               | 347 |
| Frequently asked questions .....                  | 354 |

## Overview

This section includes:

|                              |     |
|------------------------------|-----|
| Notational conventions ..... | 283 |
| Definitions .....            | 283 |
| Structural template .....    | 286 |

|                                    |     |
|------------------------------------|-----|
| Wizard page template .....         | 287 |
| General points .....               | 287 |
| Generic structure and syntax ..... | 287 |

## Notational conventions

The following notation is used to describe the structure of wizards:

### Conventions used

|               |                                                                                                                                                                                                                                           |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [ ]           | Square brackets are used to reference the value of a field in the database (in the case of contextual wizards) or one of the "special fields": "CurrentSelection" and "CurrentTable". They are also used to indicate optional parameters. |
| <<br>>        | Angle brackets are denote values for properties described in plain language. These angle brackets are not meant to be typed as is. Replace them and the text within with the appropriate information.                                     |
|               | The pipe character is used to separate possible values for a property. It is also used to separate the titles and values of multi-column lists.                                                                                           |
| { }           | Curly bracket frame the definition of a node or a block of script spanning several lines for a property. They are also used to reference the value of a wizard property.                                                                  |
| '             | In examples of Basic code, the apostrophe designates a line of comments that is not interpreted by Asset Manager.                                                                                                                         |
| ;<br>or<br>// | In the wizards, the semicolon or two slashes designate a line of comments, which is not interpreted by Asset Manager.                                                                                                                     |

## Definitions

You will find below definitions of the terms used in the description of the structure of wizards.

### Twip

The "Twip" is unit of size, and distance by default, used by the wizards. It is independent of the resolution of the screen. It corresponds to:

- 1440 "twips" equals an inch.
- 567 "twips" equals a centimeter.
- In 96 dpi resolution (standard Windows resolution) 15 twips is equivalent to 1 pixel.

## Control

A control designates a graphical element that allows you to edit a data item. Common controls are check boxes, text edits, buttons, drop-down lists, and so on.

## Node

A node corresponds to a hierarchical level in the tree-structure of the wizard. A sub-node of a given node "N", is a node one level down in the tree, attached to node "N".

**Note:** Only accented characters are not allowed in the names of nodes. The names of nodes are limited to 22 characters.

## Object

An object is a generic term that designates:

- An entire wizard.
- A wizard page.
- A control (check box, text edit, button, drop-down list, and so on) in a page.
- A variable.
- Etc.

## Parent object and child object

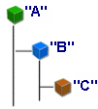
If an object "A" contains an object "B":

- Object "A" is named the parent object of object "B".
- Object "B" is named the "child object" of object "A".

**Caution:** This relationship concerns composition and not inheritance.

## Full name of an object

The full name of an object is made up of the name of all its parent objects and the name of the object itself. Each object is separated by a period ("."). Let's take the following structure as an example:



The full name of object "C" is therefore:

<Name of the object "A">.<Name of the object "B">.<Name of the object "C">

## Variable

A variable is a named storage space containing data that can be modified during the execution of a wizard. Each variable has a name allowing it to be identified uniquely within the wizard. All variables used in the wizard are global. This means that they can be referenced in any given node of the wizard using their full name.

Wizards in Asset Manager use two types of variable:

- "Wizard variables" that are defined in "LONG" or "STRING" type nodes. The type of the node defines the type of the variable; A variable defined in a "LONG" node is a long integer, a variable defined in a "STRING" node is a character string. These variables are, by definition, global. That is to say that they can be referenced using their full name from any node in the wizard. If necessary, these variables are recalculated automatically by Asset Manager.
- Basic variables, used in Basic scripts within the wizard. By default, these variables are local, but can be made global using the "COMMON" and "GLOBAL" properties. These variables are not recalculated automatically by Asset Manager.

## Transition

A transition designates what happens from one page in the wizard to another. Several transitions can be defined for a given page. Each one of these has its own user-defined conditions that determine its validity and must be fulfilled to trigger the transition:

- When the user clicks **Next**, the first valid transition is executed (that is, that for which the conditions are met). If no transitions are valid, the **Next** button is disabled.
- If the wizard has mandatory properties that are not populated, you cannot use the **Next** button.

- If the user clicks **Finish** before going through each step of the wizard in full, the default values for the unfinished steps are used.

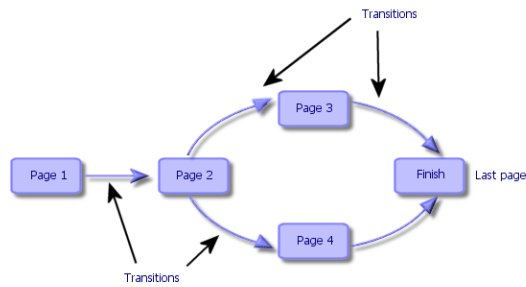
## Structural template

An Asset Manager wizard results from the execution of a wizard. By definition, the structure of the wizard is based on the structure script. I.e.:

- A wizard script (and therefore a wizard) is made up of nodes.
- Each node of the wizard has a name, one or more sub-nodes and a set of properties. The node types are listed below:
  - "ROOT" (Root node). This node is unique and encompasses all the others.
  - "START". This node is unique and contains a script that is executed when the wizard is started up.
  - "PAGE". This type of node describes a wizard page.
  - "TRANSITION". This type of node describes the transition between two "PAGE" type nodes.
  - "FINISH". This node is unique and contains a script that is executed at the end of the wizard.
  - "PARAMS". This node is unique and contains the parameters to be passed to another wizard. Several wizards can be executed in succession (with or without exchange of parameters). These wizards are said to be chained.
  - "LONG" or "STRING". This type of node defines a corresponding variable type.
- The value of a property is made explicit either via a constant or via a Basic script (in this case, the value results from the evaluation of the script).

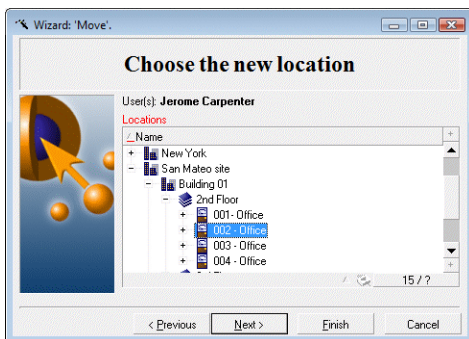
Wizards are made up of pages linked by transitions. The branching from one page to another is determined by the information entered the choices made by the user. The following diagram illustrates the structure of a wizard:

## Wizards - structural templates



## Wizard page template

A wizard page is organized as follows:



## General points

The code of a wizard (**Wizard script** field (SQL name: WizardScript) in the **Wizard** tab of the detail of an action using a wizard) takes the forms of structured text, constituted of blocks delimited by braces ({ }). This text defines the structure of a wizard.

Each node ("Root", "Page", and so on) in the tree of the wizard has an unlimited number of sub-nodes and a set of properties.

## Generic structure and syntax

Nodes have the following structure and syntax:

```
; This is a comment outside of the script
{ <Node type> <Node name>
<Name of the property> = <Value of the property>
' This is a comment inside the script
...
{ <Name of the property> =
 ...
}
{ <Node type> <Node name>
 <Name of the property> = <Value of the property>
 ...
 { <Name of the property> =
 ...
 }
}
}
```

The following rules are applied to nodes:

- The names of nodes are optional. If no name is specified for a node, Asset Manager assigns a name and number to it automatically.
- The names of nodes may not include spaces.
- If the name of a node is "=", it is no longer a node but a multi-line property. For further information on multi-line properties, Refer to "Syntax of properties" in this chapter.
- Lines beginning with a semicolon (";") outside of a script and lines beginning with an apostrophe (') within a script are interpreted as comments and are ignored.

**Note:** Attention: The space between the ("{" brace and the node type must be respected. If this is not the case, Asset Manager will refuse to execute the wizard.

## Sequencing wizards

Once executed, a wizard can be used to trigger the execution of another wizard and pass parameters (variables) to this wizard. This is known as sequencing wizards.

This section includes:

|                  |     |
|------------------|-----|
| Execution .....  | 289 |
| Parameters ..... | 289 |



## Execution

In order for a wizard A to trigger a wizard B, its Finish must not have the CHAIN property. This property must have the value of the SQL name of the **Assistant** type action to be executed, in this case "B".

**Tip:** By default, chaining of a private wizard action is not possible except for the action's owner (author). Thus users cannot chain (execute) a "hidden" wizard from an available wizard.

A private action is an action for which the **Not shared** (bPrivate) checkbox is checked. In this case, it is hidden to users apart from the owner (author).

If you wish to allow other users to chain (execute) a "hidden" wizard from an available wizard, you should use the **Administration/ Database options** menu link, and set the **Wizards/ ChainNotOwnedPrivWizards** option to **Yes**.

## Parameters

Parameters are passed to wizard B using the PARAMS node of wizard A. These parameters are added to those of the PARAMS node of wizard B. If same parameter is defined for both the PARAMS node of wizard A and in the PARAMS node of wizard B, the parameter in wizard A takes precedence over that in wizard B.

## Basic functions

In addition to the generic functions of Asset Manager (with the exception of the "AmCounter" function), wizards accept the following additional functions:

- AmComputeString()
- AmDecrementLogLevel()
- AmExecTransition()

**Caution:** This API cannot be used in Web client.

- AmLog()
- AmMsgBox()

- AmPagePath()
- AmProgress()
- AmRefreshProperties
- Am SetProperty
- AmUpdateDetail
- AmValueOf
- AmWizChain

This section includes:

|                                              |     |
|----------------------------------------------|-----|
| Values returned by the functions .....       | 290 |
| Concatenating strings in Basic scripts. .... | 290 |

## Values returned by the functions

When you call Basic functions in a wizard script, you must always assign the value returned by the function to a variable. Otherwise the Basic compiler will return an error. The following example will not compile:

```
AmGetFieldLongValue(hRecord, "lUserId", {lEmplDeptId})
```

The correct script is as follows:

```
Dim lValue as Long
lValue=AmGetFieldLongValue(hRecord, "lUserId", {lEmplDeptId})
```

## Concatenating strings in Basic scripts.

Wizard scripts can call on Basic scripts.

In Basic scripts, to concatenate text strings, use the **&** operator and not the **+** operator.

In effect, the **+** operator may be interpreted as an addition operator, which will raise an error on executing the wizard.

## Properties of a node

The values of properties can be defined using constants or scripts. Constants can be numeric, Boolean, or text.

**Note:** The properties associated with objects can be optional or mandatory. They can either be "logical" (they complement the definition of the object) or "physical" (they have an impact on the visual aspect of the object).

This section includes:

|                                                                    |     |
|--------------------------------------------------------------------|-----|
| Declarative template .....                                         | 291 |
| Defining a constant as a value for a property .....                | 292 |
| Referencing a property .....                                       | 292 |
| Defining a script as value for a property .....                    | 293 |
| Methods applicable in properties .....                             | 294 |
| Table type property .....                                          | 294 |
| Using the global variables CurrentTable and CurrentSelection ..... | 295 |

## Declarative template

A property is defined according to a declarative mode that defines circular references ( $A=\{B\}$ ,  $B=\{A\}$ ):

`<Name of the property>=<Script>`

A list of dependencies is associated with this definition. If we have:

$A=\{B\}+\{C\}$

Property "A" depends on properties "B" and "C". This list of dependencies of "A" is therefore: "B", "C".

As a result, a property changes:

- If one of the properties in its list of dependencies changes.
- Following a user action that changes a property or a dependent property.

## Defining a constant as a value for a property

The following syntaxes define a constant value for a property:

- Text type property:
  - <Name of the property> = "<Text>"
- Boolean type property:
  - <Name of the property> = TRUE
  - <Name of the property> = FALSE
  - <Name of the property> (equivalent to <Name of the property> = TRUE)
- Numeric type property:
  - <Name of the property> = 42
- <Name of the property> = {<Full name of a Basic variable or a property>}

**Note:** The Boolean value "TRUE" is equivalent to a numeric value other than "0". "FALSE" is equivalent to the numeric value "0".

## Referencing a property

To reference a property of an object (I.e. make reference to the contents of this property or object, and in particular its value), the syntax is as follows:

```
{<Full name of the property>}
```

Thus, if you want to reference the property "Prop" of a page "Page1", you write:

```
{Page1.Prop}
```

In this syntax, the full name is case insensitive.

## Defining a script as value for a property

### Notion of script

A script is a single or multi-line Basic program that returns a value in the global variable "RetVal". In the case of a single line script, this variable is implicit. In the case of a multiple-line script you must specify it.

As is the case for all Basic scripts, pay attention to the type of the returned value. This depends on the type of the property calculated via the script.

### Syntax of a single line script

```
<Name of the property>=<Script>
```

For example:

```
Variable="The name is: " & {Name}
```

The previous single line script is equivalent to the following multiple line script:

```
{ Variable =
RetVal="The name is: " & {Name}
}
```

### Syntax of a multiple-line script

```
{ <Name of the property >=
 <Script>
}
```

For example:

```
{ LABEL =
 IF {Page1.Title}="Choose an employee" THEN
RetVal="Employee"
 ELSE
RetVal="Department"
 END IF
}
```

## Methods applicable in properties

A method allows you to recover a value linked to a property or node or even execute a function on this property. In this sense, it can be considered as an advanced function.

The syntax of a method is as follows:

```
{node.node.node[.property][.method([arg1[, arg2[...]])]}
```

with:

- node: name of the node
- property: name of the property
- method: name of the method
- arg1, arg2, and so on: Basic statement or expression (this must not contain bracket characters).

**Note:** In this example, the characters "[" and "]" frame optional items.

For example, to recover the number of lines from the "LISTBOX" control in page "PAGE1", we use the "COUNT" method associated with this type of control. The command is as follows:

```
{PAGE1.LISTBOX.VALUES.COUNT()}
```

## Table type property

Table type properties are properties whose value is defined according to the following format:

```
<Column|Column|Column|...>=<Id of the line>, <Column|Column|Column|...>=<Id of the line>, ...
```

The values of these properties can be viewed in the form of a table:

|                |                                  | Column 1   | Column 2   | Column 3   |
|----------------|----------------------------------|------------|------------|------------|
| Line number: 1 | Id of the line (for example, 18) | Cell (1,1) | Cell (2,1) | Cell (3,1) |
| Line number: 2 | Id of the line (for example, 29) | Cell (1,2) | Cell (2,2) | Cell (3,2) |
| Line number: 3 | Id of the line (for example, 78) | Cell (1,3) | Cell (2,3) | Cell (3,3) |
| Etc.           | Etc.                             | Etc.       | Etc.       | Etc.       |

**Note:** The identifier is a 'Text'-type identifier.

## Example

Let's consider the "VALUES" property of the "LISTBOX" node having as its value the result of a query on the Employees and departments table. The query in question returns the values of the **Name** (SQL name: Name) and **First** (SQL name: FirstName) fields for each record in this table. Let's suppose that this property has the following value:

```
VALUES="Colombo|Gerard=32,Lubeck|Alexander=64,Daquin|William=24"
```

This value can be viewed in the form of a table:

|          |           | <b>Name</b> | <b>First name</b> |
|----------|-----------|-------------|-------------------|
| <b>1</b> | <b>32</b> | Colombo     | Gerard            |
| <b>2</b> | <b>64</b> | Lübeck      | Alexander         |
| <b>3</b> | <b>24</b> | Daquin      | William           |

## Using the global variables CurrentTable and CurrentSelection

The contents of these variables can be recovered using the following syntax:

```
[CurrentTable]
[CurrentSelection]
```

The following table presents the features of these two variables:

| <b>Name of the variable</b> | <b>Description of the variable</b>                                                                                                                                     | <b>Comment</b>                                                                              |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| CurrentTable                | Contains the SQL name of the active table at the time the wizard is launched. If there is no active table, it contains an empty string.<br><br>"String" type variable. | This variable is automatically populated by Asset Manager. The user cannot force the value. |
| CurrentSelection            | Contains the list of internal identifiers of records selected at                                                                                                       | This variable is automatically populated by Asset Manager>. It contains an empty string     |

| Name of the variable | Description of the variable                                                       | Comment                                                                                      |
|----------------------|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
|                      | the time the wizard is launched, separated by a comma.<br>"String" type variable. | if there is no selection or if no tables have been updated. The user cannot force the value. |

## Types of nodes

This section includes:

|                                       |     |
|---------------------------------------|-----|
| Root node .....                       | 296 |
| Page node .....                       | 302 |
| Transition node .....                 | 304 |
| Properties of a Transition node ..... | 305 |
| Finish node .....                     | 306 |
| Start node .....                      | 307 |
| Timer node .....                      | 308 |
| Long and String nodes .....           | 309 |
| Control node .....                    | 309 |

## Root node

### Definition of a Root node

The "Root" node describes the wizard as a whole. It is made up of a block of general properties, which can be applied to all the wizards and series of sub-nodes that represent objects contained in the wizard.

### Syntax of a Root node

The syntax of a "Root" node is as follows:

```
' Block of general properties of the root node
NAME=...
IMAGE=...
```



```

...
' Definition of sub-nodes of the root node
{ FINISH
 ...
}
{ PAGE
 ...
}
{ TRANSITION
 ...
}

```

### Properties of a Root node

The following tables list all the logical and physical properties that can be defined in a "Root" node:

#### Logical properties of the "Root" node

| Name of the property=Value     | Description of the property                                                              | Example                       | Comment                                                                                                                                                                                                                                                                                                    |
|--------------------------------|------------------------------------------------------------------------------------------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NAME="<Name of the wizard>"    | Defines the title of the window of the wizard.<br><br>"String" type property.            | NAME = "MoveWizard"           | You must define a value for this property. The name of the wizard is limited to 22 characters. This property is used to for the serialization of the wizard: The data relating to this wizard is stored under this name. As a consequence, it is preferable for different wizards to have different names. |
| TITLE="<"Title of the window>" | Defines the name of the wizard.<br><br>"String" type property.                           | TITLE=" Move wizard"          | We strongly recommend defining a value for this property.                                                                                                                                                                                                                                                  |
| GLOBAL="<Script>               | Allows you to serialize the wizard (=TRUE) or not (=FALSE). If the wizard is serialized, | {GLOBAL=Dim Filter As String} |                                                                                                                                                                                                                                                                                                            |

**Logical properties of the "Root" node, continued**

| Name of the property=Value | Description of the property                                                                                                                                                                                                                                                                                         | Example        | Comment                                      |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------------|
|                            | <p>it conserves (storing them in the ".ini" file) the values previously entered for the next time it is executed.</p> <p>This script declares, but is not executed. To execute a script at startup, use the "START" node.</p> <p>"Boolean" type property.</p>                                                       |                |                                              |
| COMMON                     | <p>Contains the Basic functions automatically included in all wizards.</p> <p>This property corresponds to the <b>gbase.wiz</b> file (read only), which is inserted into the database when it is created.</p>                                                                                                       |                |                                              |
| SERIALIZE=<TRUE FALSE>     | <p>Allows you to serialize the wizard (=TRUE) or not (=FALSE). If the wizard is serialized, it conserves (storing them in the <b>.ini</b> file) the values previously entered for the next time it is executed.</p> <p>The NAME property determines in which section of the <b>.ini</b> file the values will be</p> | SERIALIZE=TRUE | By default, this property is set to "FALSE". |

**Logical properties of the "Root" node, continued**

| Name of the property=Value | Description of the property                                                                                                                                                      | Example | Comment                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|------------------------------------------------------------------------------------------------------------|
|                            | <p>stored.</p> <p>"Boolean" type property.</p> <p>Location of the .ini files: see <b>Asset Manager - Installation and upgrade</b> guide, chapter <b>.ini and .cfg files</b>.</p> |         |                                                                                                            |
| MODAL=<TRUE FALSE>         | Defines whether the wizard is modal (=TRUE) or not (=FALSE).                                                                                                                     |         | As by design, if you have multi-tenancy enabled, non-modal wizards are forced to operate as modal wizards. |

**Physical properties of the "Root" node**

| Name of the property=Value                                            | Description of the property                                                                                        | Example                      | Comment                                                                                                                                                                                         |
|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IMAGE="<Path of the bitmap file> "IMAGE16="<Path of the bitmap file>" | <p>Defines the bitmap type graphical file (.bmp) to be displayed in the wizard.</p> <p>"String" type property.</p> | IMAGE="C:\Images\Page 1.bmp" | <p>If no value is defined for this property, then no image will be displayed. The path of the graphics file depends on the Asset Manager Config folder.</p> <p>Asset Manager first searches</p> |

**Physical properties of the "Root" node, continued**

| Name of the property=Value | Description of the property                                                                                                                                                                            | Example     | Comment                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            |                                                                                                                                                                                                        |             | <p>for the<br/>           imagine in<br/>           the<br/>           database</p> <p>If you<br/>           define a<br/>           value for<br/>           "IMAGE1<br/>           6", this<br/>           property is<br/>           used<br/>           instead of<br/>           "IMAGE"<br/>           when the<br/>           color depth<br/>           of the<br/>           screen is<br/>           16.</p> |
| WIDTH=<Width>              | <p>Defines the default<br/>           width ("&lt;Width&gt;") of<br/>           the window of the<br/>           wizard. This is<br/>           expressed in twips.</p> <p>"Long" type property.</p>   | WIDTH=6000  |                                                                                                                                                                                                                                                                                                                                                                                                                        |
| HEIGHT=<Height>            | <p>Defines the default<br/>           height ("&lt;Height&gt;") of<br/>           the window of the<br/>           wizard. This is<br/>           expressed in twips.</p> <p>"Long" type property.</p> | HEIGHT=5000 |                                                                                                                                                                                                                                                                                                                                                                                                                        |
| MINWIDTH=<MinWidth>        | <p>Defines the minimum<br/>           width of the wizard<br/>           window.</p> <p>Value expressed in<br/> <b>twips.</b></p>                                                                      |             |                                                                                                                                                                                                                                                                                                                                                                                                                        |
| MINHEIGHT=<MinHeight>      | <p>Defines the maximum<br/>           height of the wizard<br/>           window.</p>                                                                                                                  |             |                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Physical properties of the "Root" node, continued**

| Name of the property=Value  | Description of the property                                                                                                                                           | Example | Comment |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------|
|                             | Value expressed in <b>twips</b> .                                                                                                                                     |         |         |
| CTRLHEIGHT=<CtrlHeight>     | <p>Defines the height of a control whose vertical size is set (for example, a TEXTBOX control).</p> <p>Spacing value expressed in <b>twips</b>.</p>                   |         |         |
| LABELSPACING=<labelSpacing> | <p>Defines the space between the label of a control and the control itself (when the title is above the control).</p> <p>Spacing value expressed in <b>twips</b>.</p> |         |         |
| CTRLSPACING=<CtrlSpacing>   | <p>Defines the vertical spacing between two controls.</p> <p>Spacing value expressed in <b>twips</b>.</p>                                                             |         |         |
| IMGBORDER=<Width>           | <p>Defines the horizontal spacing between the wizard image and its controls.</p> <p>Value expressed in <b>twips</b>.</p>                                              |         |         |
| NAVIGATION=<TRUE FALSE>     | <p>Displays (=TRUE) or not (=FALSE) that navigation bar containing the <b>Next</b> and <b>Cancel</b> buttons in the wizard window.</p>                                |         |         |
| CONFIRMCANCEL=<TRUE FALSE>  | <p>Displays (=TRUE) or not (=FALSE) the message confirming the cancellation.</p>                                                                                      |         |         |

### Physical properties of the "Root" node, continued

| Name of the property=Value | Description of the property                                                                                                                | Example | Comment |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|---------|---------|
| DEFAULTONNEXT=<TRUE FALSE> | <p>Selects by default (=TRUE) the <b>Next</b> button.</p> <p>If DEFAULTONNEXT=FALSE then the button selected by default is <b>End</b>.</p> |         |         |

### Sub-nodes of a Root node

The types of sub-nodes that you can define for a root node are listed in the table below. Each type of node represents an "Object".

#### Sub-nodes of "Root" node

| Node type | Description                                                                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PAGE      | Describes a page of the wizard.                                                                                                                                       |
| FINISH    | Describes the final transition from the final page of the wizard (to the finish). This "Transition" type node does not have "FROM" and "TO" properties.               |
| START     | Contains, for example, a script to be executed when the wizard is launched (using the "DO" property) and the name of the starting page of the wizard ("TO" property). |
| PARAMS    | Enables you to transfer parameters from one wizard to another (if the CHAIN property of the FINISH sub-node is populated).                                            |
| TIMER     | Enables you to associate a timer with a wizard page.                                                                                                                  |

## Page node

### Definition of a Page node

A "Page" node describes a page in a wizard. It is made up of a block of properties applicable to this node and all its sub-nodes; and a set of sub-nodes that define objects defined in the page.

## Syntax of a Page node

The syntax of a "Page" node is as follows:

```
' Declaration of the page
{ Page <Name of the page>
' Block properties of the page node
IMAGE=...
TITLE=...
' Definition of sub-nodes of the "Page" node
{ TRANSITION
 ...
}
{ <Control type> <Control name>
 ...
}
 ...
}
```

## Properties of a Page node

The following tables list all the properties that can be defined in a "Page" node:

### Logical properties of a "Page" node

| Name of the property=Value  | Description of the property                                                                                                                    | Example                        | Comment                                                                                                                                                       |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TITLE="<Title of the page>" | Defines the title of the page. This title appears in bold at the top of the page.<br><br>"String" type property.                               | TITLE="Move"                   | If no value is defined for this property, it inherits the value of the "TITLE" property of the "Root" node. Unlike labels, this string does not support HTML. |
| ONENTER=<Script>            | Defines a Basic script that is executed when the page is accessed by clicking <b>Next</b> or <b>Previous</b> .<br><br>"Boolean" type property. | {ONENTER = AmMsgBox ("Hello")} |                                                                                                                                                               |

### Physical properties of a "Page" node

| Name of the property=Value                                                   | Description of the property                                                                                              | Example                          | Comment                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IMAGE="<Path of the bitmap file>"<br><br>IMAGE16="<Path of the bitmap file>" | Defines the bitmap type graphical file (.bmp) to be displayed in the page of the wizard.<br><br>"Boolean" type property. | IMAGE ="<br>C:\Images\Page1.bmp" | If no value is defined for this property, it inherits the value of the "IMAGE" property of the "Root" node.<br><br>If you define an empty value for this property, no image is displayed.<br><br>If you define a value for "IMAGE16", this property is used instead of "IMAGE" when the color depth of the screen is 16. |

### Sub-nodes of a Page node

Two types of sub-nodes can be defined for a "Page" node:

#### Sub-nodes of "Page" node

| Node type / "Object"                | Description                                                                    |
|-------------------------------------|--------------------------------------------------------------------------------|
| <Control type> <Control name>       | Defines a control displayed in the current page.                               |
| TRANSITION <Name of the transition> | Describes a transition between the current page and another page of the wizard |
| TIMER                               | Enables you to associate a timer with a wizard page.                           |

## Transition node

### Definition of a Transition node

A "Transition" node describes the passage between two pages in a wizard. It is exclusively made up of a block of properties.

**Note:** Transitions can be defined from the inside of a "Page" node (in this case, they do not require the "FROM" property) or in the "Root" node. The final transition leading to the closure of the wizard, is described in a "FINISH" node (at "Root" node level) and does not have "FROM" and "TO"



properties.

## Syntax of a Transition node

The syntax of a "Transition" node is as follows:

```
' Declaration of the transition
{ TRANSITION0 <Name of the transition>
' Block of properties of the transition node
FROM=...
TO=...
CONDITION=...
}
```

## Properties of a Transition node

The following table lists all the properties that can be defined in a "Transition" node:

### Logical properties of a "Transition" node

| Name of the property=Value         | Description of the property                                                                                 | Example                        | Comment                                                                                                                                                      |
|------------------------------------|-------------------------------------------------------------------------------------------------------------|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FROM="<Name of the original page>" | Defines the name of the original page of the transition.<br><br>"String" type property.                     | FROM="Page2"                   | This property is mandatory if the transition is defined inside the "Root" node and inapplicable if it is defined in a "Page" node, "Finish" or "Start" node. |
| TO="<Name of the target page>"     | Defines the name of the target page of the transition.<br><br>"String" type property.                       | TO="Page3"                     | This property is mandatory if the transition is defined inside a "Root" or "Page" node and inapplicable if it is defined in a "Finish" node.                 |
| CONDITION=<Script>                 | Defines the condition to be fulfilled to cause the transition.<br><br>"Script" type property that returns a | CONDITION={Comment}<br>="user" | This property is not available in "Start" nodes.                                                                                                             |

**Logical properties of a "Transition" node, continued**

| Name of the property=Value | Description of the property                                                                    | Example         | Comment |
|----------------------------|------------------------------------------------------------------------------------------------|-----------------|---------|
|                            | Boolean.                                                                                       |                 |         |
| DO=<Script>                | Defines a script to be executed at the time of the transition.<br><br>"Boolean" type property. | {DO= Filter=""} |         |

**Specificities of a Transition node**

A "Transition" node does not have a sub-node.

**Why define transitions in the "Root" node?**

Taking transitions out of "Page" nodes allows you to create pages that can be used in any scripts and rationalizes the writing of scripts.

## Finish node

A "Finish" node describes the final transition: that which leads to the final page of the wizard. It is a specific type of "Transition" node that does not have "FROM" and "TO" properties. Apart from this exception, the syntax and properties in a "Finish" node are identical to those of a "Transition" node.

The CHAIN property, specific to the Finish node allows you to trigger the execution of another wizard.

**Logical properties of a "Finish" node**

| Name of the property=Value                    | Description of the property                                                                                                                                                              | Example        | Comment |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|---------|
| CHAIN=<SQL name of the wizard to be executed> | Defines the SQL name of the wizard to be executed at the end of the current wizard.<br><br>If this property is not populated, no wizards will be executed.<br><br>"String" type property | CHAIN = "Move" |         |
| CONDITION=<Script>                            | Defines the condition validating the <b>Finish</b> button.<br><br>"Boolean" type property.                                                                                               |                |         |

### Logical properties of a "Finish" node, continued

| Name of the property=Value | Description of the property                                                                                   | Example | Comment |
|----------------------------|---------------------------------------------------------------------------------------------------------------|---------|---------|
| DO=<Script>                | Defines the script to execute at the end of the wizard.<br><br>"Script" type property that returns a Boolean. |         |         |

**Note:** The PARAMS node allows you to pass parameters to the following wizard.

### Physical property of a "Finish" node

| Name of the property=Value   | Description of the property                                                                                                                            | Example |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| SUMMARY=<TRUE FALSE>         | Displays (=TRUE) or not (=FALSE) a summary page when launching the wizard.<br><br>The amLog and amProgress functions enable you to populate this page. |         |
| SHOWPROGRESSBAR=<TRUE FALSE> | Displays (=TRUE) or not (=FALSE) a progress bar in the summary page.                                                                                   |         |
| SHOWLOGLIST=<TRUE FALSE>     | Displays (=TRUE) or not (=FALSE) a progress log in the summary page.                                                                                   |         |
| LABEL="Title"                | Displays a title for the summary page.                                                                                                                 |         |
| ISHTML=<TRUE FALSE>          | Defines the nature of the text of the caption.<br><br>"Boolean" type property.                                                                         |         |
| TITLE="Title"                | Defines a title for the summary page.<br><br>The default title of the summary page is that of the root node.                                           |         |

## Start node

A "Start" node describes how the wizard is started. It is a specific type of "Transition" node that does not have a "FROM" or "CONDITION" property. Apart from this exception, the syntax and properties in a "Start" node are identical to those of a "Transition" node.

### Logical property of a "Start" node

| Name of the property=Value       | Description of the property                                                                  | Example |
|----------------------------------|----------------------------------------------------------------------------------------------|---------|
| DO=<Script>                      | Defines the script to execute at start-up.<br>"Script" type property that returns a Boolean. |         |
| TO="<Name of the starting page>" | Defines the name of the first page to display.<br>"String" type property.                    |         |

**Note:** If this node does not exist, the wizard starts on the first page.

## Timer node

A "Timer" node enables you to perform a task at a regular interval.

### Logical property of a "Timer" node

| Name of the property=Value | Description of the property                                                                                                                                                                          | Example |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| AUTO=<TRUE FALSE>          | Defines whether the timer is automatically launched when the page is displayed.<br>"Boolean" type property.                                                                                          |         |
| ENABLED=<TRUE FALSE>       | Defines whether the timer is enabled (=TRUE) or not (=FALSE).<br>This property can be used to stop or restart a timer.<br>"Boolean" type property.                                                   |         |
| INTERVAL=time interval     | Defines the time interval between two executions of the timer.<br>Duration expressed in milliseconds.                                                                                                |         |
| TIMER="Script"             | Defines what must be executed each time the timer interval has passed.<br>Script type property:                                                                                                      |         |
| VALUE=tickcount            | Number of times the interval has been passed. Each property dependent on this interval will be automatically reevaluated at each occurrence.<br>This value (tickcount) is automatically incremented. |         |

## Long and String nodes

Long and String nodes define variables. These can be referenced in all nodes of the wizard. The name of the node determines the name of the variable.

These nodes only have one single property whose type depends on the node; its type is LONG for a Long node and STRING for a String node. This property, VALUE, allows you to define the value of the variable.

### Logical property of a Long or String node

| Name of the property=Value | Description of the property                                                                                                                  | Example  | Comment |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|----------|---------|
| VALUE=<Value>              | Defines the value of the variable whose name is that of the node.<br><br>"Long" type property for a Long node or "String" for a String node. | VALUE=12 |         |

**Note:** Long and String nodes can be defined in any node in the wizard. They do not have sub-nodes.

## Control node

### Definition of a Control node

Controls of a page interact with the user. You may define as many controls as you want for a given page. Asset Manager fully manages the organization of controls within a page. You do not have to specify the positioning of each of the controls that you define.

"Control" type nodes are exclusively made up of a block of properties applicable to a defined control.

### General syntax of a Control node

The general syntax of a "Control" type node is as follows:

```
' Declaration of the control
{ <Control type> <Control name>
' Properties of the control
```

```
...
}
```

## Types of controls and associated properties

All the controls have properties in common. However, there are properties that are specific to certain controls.

This section includes:

|                                 |     |
|---------------------------------|-----|
| Common properties .....         | 311 |
| The CHECKBOX control .....      | 314 |
| The COMBOBOX control .....      | 314 |
| The OPTIONBUTTONS control ..... | 315 |
| The LISTBOX control .....       | 316 |
| The LABEL control .....         | 321 |
| The PROGRESSBAR control .....   | 321 |
| The COMMANDBUTTON control ..... | 322 |
| The DBLISTBOX control .....     | 323 |
| The DBQUERYBOX control .....    | 326 |
| The DBEDIT control .....        | 329 |
| The DBTABLE control .....       | 330 |
| The DBPATH control .....        | 330 |
| The LINKEDIT control .....      | 330 |
| The TEXTBOX control .....       | 332 |
| The CHART control .....         | 332 |
| The FILEEDIT control .....      | 335 |
| The TICKEDIT control .....      | 335 |
| The CALENDAR control .....      | 336 |
| The TIMSPANEDIT control .....   | 336 |
| The NUMBOX control .....        | 336 |
| The COMBOEDIT control .....     | 337 |
| The DATETIMEEDIT control .....  | 337 |

## Common properties

The following table lists optional properties applicable to all controls:

### Logical properties common to all controls

| Name of the property=Value | Description of the property                                                                                                                                                                                                                                                                                                                                | Example                                                                                                                                                                                                                                                          | Comment                                                                                                                                                                                                               |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MANDATORY=<TRUE FALSE>     | <p>Forces the user to populate the control to validate a transition.</p> <p>This property is ignored if the control is:</p> <ul style="list-style-type: none"> <li>• not visible</li> <li>• read only</li> <li>• disabled</li> </ul>                                                                                                                       | MANDATORY=TRUE                                                                                                                                                                                                                                                   | This property is not available for "CHECKBOX" and "LABEL" controls                                                                                                                                                    |
| VALUE="<Value, Value,...>" | <p>Defines the default value of the control when it is created. When the control is displayed in the wizard, this value is replaced with the value that was selected by the user. Depending on the control, the VALUE property may contain several values.</p> <p>The type of this property depends on the type of control (Boolean, text, and so on).</p> | <ul style="list-style-type: none"> <li>• For a "CHECKBOX" control, &lt;Value&gt; can either be "TRUE" or "FALSE".</li> <li>• For a "LISTBOX" control where VALUE= "1,3,4"; 1,3 and 4 are values that correspond to the lines selected in the control.</li> </ul> | <p>When the VALUE property can contain several values, the control's MULTISEL property must be activated (equal to TRUE). Otherwise, only the value defined in first position of the VALUE property will be used.</p> |
| PERMANENT=<TRUE FALSE>     | When you go from one wizard page to the next, the controls are                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                       |

**Logical properties common to all controls, continued**

| Name of the property=Value | Description of the property                                                                                                                                                                       | Example | Comment                                                                                    |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|--------------------------------------------------------------------------------------------|
|                            | <p>destroyed.</p> <p>Defines if the control is kept and hidden when going from one page to another (=TRUE) instead of being destroyed (=FALSE).</p> <p>"Boolean" type property.</p>               |         |                                                                                            |
| SERIALIZE=<TRUE FALSE>     | <p>This property enables you to serialize the wizard at the level of the control. If the serialization of the root node is active (=TRUE), you can deactivate it at the level of the control.</p> |         | <p>By default, this property has the value of the SERIALIZE property at the root node.</p> |
| HELP = "help"              | <p>This property enables you to include help text in HTML format into the control of a wizard. You access this help by pressing SHIFT+F1.</p>                                                     |         |                                                                                            |

**Physical properties common to all controls**

| Name of the property=Value | Description of the property                                                                            | Example               | Comment |
|----------------------------|--------------------------------------------------------------------------------------------------------|-----------------------|---------|
| VISIBLE=<TRUE FALSE>       | <p>Defines whether the control is visible (=TRUE) or not (=FALSE).</p> <p>"Boolean" type property.</p> | Label1.Visible=TRUE   |         |
| ENABLED = <TRUE FALSE>     | <p>Defines whether the control is active (=TRUE) or not</p>                                            | Choice1.Enabled=FALSE |         |



**Physical properties common to all controls, continued**

| Name of the property=Value                 | Description of the property                                                                                                                                                            | Example                              | Comment                         |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|---------------------------------|
|                                            | <p>(=FALSE). "Boolean" type property.</p> <p>"Boolean" type property.</p>                                                                                                              |                                      |                                 |
| <p>READONLY = &lt;TRUE FALSE&gt;</p>       | <p>Defines whether the value of the control is in read-only (=TRUE) and thus cannot be modified b the user, or whether it is editable (=FALSE).</p>                                    | <p>READONLY=TRUE</p>                 |                                 |
| <p>LABEL = "&lt;Text of the label&gt;"</p> | <p>Defines an optional text, displayed above or to the left of the control.</p> <p>"String" type property.</p>                                                                         | <p>Choice1.Label="Select person"</p> | <p>This label supports HTML</p> |
| <p>LABELLEFT</p>                           | <p>This property enables you to put the control's label on its left.</p> <p>Using this property means that you must populate the XOFFSET property.</p> <p>"Boolean" type property.</p> |                                      |                                 |
| <p>XOFFSET</p>                             | <p>Defines the space reserved for the control's label if it has been placed to the left of the control (by using the LABELLEFT property).</p> <p><b>Twip</b> type property.</p>        |                                      |                                 |
| <p>ISHTML</p>                              | <p>Defines the nature of the text of the label.</p> <p>By default, the nature of the test is HTML.</p> <p>"Boolean" type property.</p>                                                 |                                      | <p>This label supports HTML</p> |
| <p>INDENT</p>                              | <p>This property enables you to shift the control and its caption to the right.</p> <p><b>Twip</b> type property.</p>                                                                  |                                      |                                 |

### Physical properties common to all controls, continued

| Name of the property=Value | Description of the property                                                           | Example | Comment |
|----------------------------|---------------------------------------------------------------------------------------|---------|---------|
| YOFFSET                    | Defines an 'offset' before the control and its caption.<br><b>Twip</b> type property. |         |         |
| YOFFSET2                   | Defines a 'offset' after the control and its caption.<br><b>Twip</b> type property.   |         |         |

## The CHECKBOX control

The "CHECKBOX" control defines a check box.

### Properties

In addition to the optional properties common to all controls, the "CHECKBOX" control recognizes the following property:

#### Property of "CHECKBOX" control

| Name of the property=Value | Description of the property                                                                                                            | Example                 |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| CAPTION="<Text>"           | Defines the text of the check box. "String<br>This text cannot be in HTML and must contain one single line.<br>"String" type property. | TEXT="Identify by name" |

## The COMBOBOX control

The "COMBOBOX" control defines a single choice in an enumeration (itemized list) of predefined values.

### Properties

In addition to the optional properties common to all controls, the "COMBOBOX" control recognizes the

following property:

**Physical properties of the "COMBOBOX" control**

| Name of the property=Value                        | Description of the property                                                                                                                                                                                                               | Example                                   | Comment                                                                                                                                                                |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VALUES="<Name=Value, Name=Value, Name=Value,...>" | <p>Defines the value couples ("Name"= "Value") for the "Combo" control. "Name" defines the text that is displayed in the control, "Value" the value attributed if this "Name" is selected by the user.</p> <p>"String" type property.</p> | VALUES="Table of assets=asset, User=user" | <p>If you omit the "Value", Asset Manager will automatically assign one.</p> <p><b>Example</b></p> <p>VALUES = "A,B,C" is equivalent to<br/>VALUES = "A=1,B=2,C=3"</p> |

## The OPTIONBUTTONS control

The "OPTIONBUTTONS" control defines a group of option buttons (radio buttons).

### Properties

In addition to the optional properties common to all controls, the "OPTIONBUTTONS" control recognizes the following properties:

**Physical properties of the "OPTIONBUTTONS" control**

| Name of the property=Value                            | Description of the property                                                                                                                                                                                                                          | Example                                   |
|-------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|
| VALUES="<Title=Value, Title=Value, Title=Value,....>" | <p>Defines the value couples ("Title"="Value") for the "CHOICE" control. "Name" defines the text of the option button, "Value" the value attributed to the control if this option button is selected by the user.</p> <p>"String" type property.</p> | VALUES="Table of assets=asset, User=user" |
| BORDER=<TRUE FALSE>                                   | Specifies whether a border is drawn for the group of option buttons (=TRUE) or not (=FALSE)                                                                                                                                                          | BORDER= TRUE                              |

**Physical properties of the "OPTIONBUTTONS" control, continued**

| Name of the property=Value | Description of the property                                                                                                                                                   | Example |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
|                            | If the group of buttons is framed, the text is integrated into the border of the frame. This text cannot be in HTML or contain multiple lines<br><br>"Boolean" type property. |         |

## The LISTBOX control

This "LISTBOX" control defines a list of objects that can be selected. "LISTBOX" controls can be multi-column controls.

### Properties

In addition to the optional properties common to all controls, the "LISTBOX" control recognizes the following properties:

**Physical properties of the "LISTBOX" control**

| Name of the property=Value | Description of the property                                                                                                          | Example       | Comment                                                                                                                                                  |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| LISTHEIGHT = <Percentage>  | Defines size of the "LISTBOX" control in relation to other "LISTBOX" controls in the wizard as a whole.<br><br>"Long" type property. | LISTHEIGHT=50 | If there are two "LISTBOX" controls with values "10" and "20" respectively for this property, the second control will be twice as high as the first one. |
| MULTISEL = <TRUE FALSE>    | Specifies whether the control supports multiple-                                                                                     | MULTISEL=TRUE |                                                                                                                                                          |

**Physical properties of the "LISTBOX" control, continued**

| Name of the property=Value                                               | Description of the property                                                                                                        | Example                                             | Comment                                                                                                                                                            |
|--------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                          | <p>selection (=TRUE) or not (=FALSE).<br/><br/>"Boolean" type property.</p>                                                        |                                                     |                                                                                                                                                                    |
| <p>COLTITLE = "&lt;Column Column Column...&gt;"</p>                      | <p>Defines the title of the columns in the list. Replace "Column" by the title of the column.<br/><br/>"String" type property.</p> | <p>COLTITLE = "Name FirstName"</p>                  |                                                                                                                                                                    |
| <p>COLWIDTH = "&lt;Width Width Width...&gt;"</p>                         | <p>Defines the size of the columns proportionally to the overall size of the control.<br/><br/>"String" type property.</p>         | <p>COLWIDTH = "50 50"</p>                           |                                                                                                                                                                    |
| <p>VALUES = "&lt;Text Text ...= Value, Text Text ...= Value,...&gt;"</p> | <p>Defines value couples ("Text Text ..=" "Value") for the "LISTBOX" control. "Text Text .." defines the text to</p>               | <p>VALUES="Table of assets=asset, , User=user,"</p> | <p>If you omit the "Value", Asset Manager will automatically assign one.<br/><br/>VALUES="A,B,C" is equivalent to VALUES="A=1,B=2,C=3"<br/><br/><b>Example</b></p> |

**Physical properties of the "LISTBOX" control, continued**

| Name of the property=Value  | Description of the property                                                                                                                                                                      | Example            | Comment                                                                                                                                                                                                                                                                          |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | <p>be displayed in each of the columns for a line of the "LISTBOX" control, "Value" the value attributed to the control if this line is selected by the user.</p> <p>"String" type property.</p> |                    | <p>This property can be populated directly or using the AmdbGetList, function by writing, for example:</p> <p>VALUES = AmDbGetList ("SELECT Name, FirstName FROM amEmplDept WHERE Name Like 'A%', ' ', ', ', '=')</p> <p>Do not confuse the "VALUES" and "VALUE" properties.</p> |
| EDITABLE="<0 1>"            | <p>Defines whether the text in the columns can be edited or not.</p> <p>"String" type property.</p>                                                                                              | EDITABLE="0 1"     |                                                                                                                                                                                                                                                                                  |
| TABLE="<Name of the table>" | <p>Defines the application context of the column titles if they have been defined for the "COLNAME" property.</p>                                                                                | TABLE="amEmplDept" | <p><b>Note:</b> Add corresponding SQL name of the table if you want to make certain columns editable.</p> <p>For example, define TABLE="amEmplDept" if your LISTBOX displays</p>                                                                                                 |

**Physical properties of the "LISTBOX" control, continued**

| Name of the property=Value              | Description of the property                                                                                                                                                                                                                                                                                                                                                                                 | Example                                | Comment                                                                                                                                                 |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                         |                                                                                                                                                                                                                                                                                                                                                                                                             |                                        | <p>columns from the Employees and departments table.</p>                                                                                                |
| <p>COLNAME="&lt;name name name&gt;"</p> | <p>Defines the title of the properties of the columns in the list with the SQL field names. The "TABLE" property must be populated;</p> <p>Replace "name" by the SQL name of the field for the title of the column.</p> <p>If the title of the column has been populated with the "COLTITLE" property, this takes priority over the "COLNAME" property, but it keeps the type of the field (text, date,</p> | <p>COLNAME="Name FirstName dtHire"</p> | <p><b>Caution:</b> The COLNAME property <b>must</b> be populated for the content of the LISTBOX control to be properly displayed by the Web client.</p> |

**Physical properties of the "LISTBOX" control, continued**

| Name of the property=Value | Description of the property                                                    | Example    | Comment |
|----------------------------|--------------------------------------------------------------------------------|------------|---------|
|                            | and so on).                                                                    |            |         |
| MULTISEL = <TRUE FALSE>    | Defines the use of multiselecti on for a list.<br><br>"Boolean" type property. | MULTISEL=1 |         |

**Methods of the "LISTBOX" control**

| Name of the property=Value | Description of the property                                                                                                | Example                                                                                                            | Comment |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|---------|
| VALUES (i)                 | Returns the contents of the line (i).                                                                                      | a b c                                                                                                              |         |
| VALUES.COUNT()             | Calculates the number of lines in the "VALUES" property.                                                                   | retval = {listbox1.values.count()}                                                                                 |         |
| VALUES.CELL (h,v)          | Returns the contents of the cell designated by its coordinates (horizontal, vertical).                                     | VALUES.CELL(2,4)                                                                                                   |         |
| VALUES.COLUMN (i)          | Returns the contents (value) of the column (i).<br><br>If i=0 or is empty, this instruction returns the IDs of the column. | VALUES.COLUMN(1)                                                                                                   |         |
| VALUES.LINE(i)             | Returns the contents (value) of the line (i).<br><br>If i=0 or is empty, this instruction returns the IDs of the row.      | VALUES.LINE(1)                                                                                                     |         |
| VALUES.SORT (iCol, bAsc)   | Sorts the column (i) in an ascending order or not (bAsc=1 or bAsc=0).                                                      | { LISTBOX lb VALUES = "first,second,third" }<br><br>{ COMMANDBUTTON btn { CLICK = RetVal = {lb.Values.Sort(1)} } } |         |
| VALUES (i,0)               | Returns the value of the ID of the line (i).                                                                               |                                                                                                                    |         |



### Mandatory logical property of the "LISTBOX" control

| Name of the property=Value | Description of the property                                                                                                                                                                                                                                                                                                                                                                                                      | Example                  |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| TABLE=<Name of the table>  | Name of table used to extract the titles of columns.<br><br>"String" type property.                                                                                                                                                                                                                                                                                                                                              | TABLE= <b>amAsset</b>    |
| COLNAME=<Title Title ...>  | Defines the titles of columns, using the SQL name of fields in the table defined using the "TABLE" property. This property also allows you to define the edit controls used. The control is the same as is used in Asset Manager to populate the field.<br><br>Asset Manager will first take the values of the "COLTITLE" property (if this exists) into account to define the titles of columns.<br><br>"String" type property. | COLNAME="Name FirstName" |

## The LABEL control

The "LABEL" control simply defines a label. This control has the following property:

### Physical properties of the "LABEL" control

| Name of the property=Value | Description of the property               | Example                     | Comment |
|----------------------------|-------------------------------------------|-----------------------------|---------|
| CAPTION=<Text>             | Contains the text displayed in the label. | CAPTION="Select a location" |         |

## The PROGRESSBAR control

The "PROGRESSBAR" control defines a progress bar.

## Properties

In addition to the optional properties common to all controls, the "PROGRESSBAR" control recognizes the following properties:

### Physical properties of the "PROGRESSBAR" control

| Name of the property=Value | Description of the property                                                                                                                                               | Example      |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| MAXVALUE=<Maximum value>   | Defines the maximum value corresponding to 100% of the progress bar.<br><br>The "VALUE" property indicates the current value of the control.<br><br>"Long" type property. | MAXVALUE=200 |

## The COMMANDBUTTON control

The "COMMANDBUTTON" control defines an action button.

## Properties

In addition to the optional properties common to all controls, the "COMMANDBUTTON" control recognizes the following properties:

### Physical properties of the "COMMANDBUTTON" control

| Name of the property=Value | Description of the property                                                             | Example         |
|----------------------------|-----------------------------------------------------------------------------------------|-----------------|
| WIDTH=<Width>              | Defines in twips the width of the button.<br><br>"Long" type property.                  | WIDTH=250       |
| HEIGHT=<Height>            | Defines in twips the height of the button.<br><br>"Long" type property.                 | HEIGHT=125      |
| CAPTION=<Text>             | Defines the text (non HTML) displayed within the button.<br><br>"String" type property. | CAPTION="Start" |
| CLICK=<Basic script>       | Defines the Basic script that is executed when the user clicks the button.              |                 |

## The DBLISTBOX control

The "DBLISTBOX" control defines a list of records that can be selected from the database. This control can be a multiple-column control. The list displayed in the control is the result of a partial AQL query (only the WHERE clause is used) on the Asset Manager database.

**Note:** The "VALUE" property returns the list of identifiers ("Id") of the selected lines. You cannot access the values of cells in the list. For this, you need either to perform another query, or use a "LISTBOX" type control.

### Properties

In addition to the optional properties common to all controls, the "DBLISTBOX" control recognizes the following properties:

#### Physical properties of the "DBLISTBOX" control

| Name of the property=Value                                                  | Description of the property                                                                                                | Example                    | Comment                     |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|----------------------------|-----------------------------|
| TABLE="<SQL name of the table>"                                             | Defines the table concerned by the query.<br><br>"String" type property.                                                   | TABLE=amAsset              | This property is mandatory. |
| COLNAME="<SQL name of the field or link SQL name of the field or link ...>" | Defines the data items to be extracted from the database (identified using their SQL name).<br><br>"String" type property. | COLNAME = "Name FirstName" |                             |
| COLWIDTH = "<Width Width Width ...>"                                        | Defines the width of the columns, as a percentage of the overall size of the "DBLISTBOX" control.                          | COLWIDTH="40 60"           |                             |

**Physical properties of the "DBLISTBOX" control, continued**

| Name of the property=Value | Description of the property                                                                                                                        | Example                                       | Comment                                                                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                            | "String" type property.                                                                                                                            |                                               |                                                                                                                                                            |
| LISTHEIGHT = <Percentage>  | Defines the size of the "DBLISTBOX" control in relation to other "DBLISTBOX" controls in the wizard as a whole.<br><br>"Long" type property.       | LISTHEIGHT=50                                 | If there are two "DBLISTBOX" controls with values "10" and "20" respectively for this property, the second control will be twice as high as the first one. |
| TREE=<TRUE FALSE>          | Displays the data in tree mode (=TRUE) or not (=FALSE).<br><br>"Boolean" type property.                                                            | TREE=TRUE                                     | By default, this property is set to "FALSE"                                                                                                                |
| MULTISEL = <TRUE FALSE>    | Specifies whether the control supports multiple-selection (=TRUE) or not (=FALSE).<br><br>"Boolean" type property.                                 | MULTISEL=TRUE                                 |                                                                                                                                                            |
| DBLCLICK = <TRUE FALSE>    | If this property is set to TRUE, Asset Manager will simulate clicking the <b>Next</b> button of the current page each time you double-click a row. | DBLCLICK=FALSE                                |                                                                                                                                                            |
| FILTER= "<Condition>"      | Defines the AQL "WHERE" condition to filter records to be processed in the                                                                         | FILTER= "User.IEmpIDeptId='Colombo, Gerard' " |                                                                                                                                                            |

**Physical properties of the "DBLISTBOX" control, continued**

| Name of the property=Value         | Description of the property                                                                                                                                                                                                                   | Example                                                                                                                                  | Comment |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|---------|
|                                    | <p>query.</p> <p>"String" type property.</p>                                                                                                                                                                                                  |                                                                                                                                          |         |
| <p>MAXSEL = &lt;TRUE FALSE&gt;</p> | <p>Defines the possibility of selecting more than 99 objects (=TRUE) or not (=FALSE).</p> <p>The selection is limited to 99 objects by default (=TRUE).</p>                                                                                   |                                                                                                                                          |         |
| <p>SORT(iCol, bAsc)</p>            | <p>Sorts the column (iCol) in an ascending order or not (bAsc=1 or bAsc=0).</p>                                                                                                                                                               | <p>SORT(2, 0)</p>                                                                                                                        |         |
| <p>[Value.]ISSELECTION()</p>       | <p>If the selection contains values other than 0 (in other words, the user did not make a selection in the list), this method returns a non-null value.</p>                                                                                   | <p>MANDATORY = not {dblistbox1.IsSelection()}</p> <p>(A transition is impossible if the user has not selected anything in the list.)</p> |         |
| <p>TABLE.LABEL ([iNameType])</p>   | <p>This method returns the name of the table on which the control is defined.</p> <p>The name types are the following:</p> <ul style="list-style-type: none"> <li>• 1 - System name</li> <li>• 2 - SQL name</li> <li>• 3 - Default</li> </ul> | <p>RetVal = {dblistbox1.table.label(2)}</p>                                                                                              |         |

**Physical properties of the "DBLISTBOX" control, continued**

| Name of the property=Value | Description of the property                                                                                                               | Example | Comment |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|---------|---------|
|                            | label <ul style="list-style-type: none"> <li>4 - Description (text in help)</li> </ul> The "TABLE" property needs to have been populated. |         |         |

## The DBQUERYBOX control

The "DBQUERYBOX" control defines a list of records that can be selected. This control can be a multi-column control. The list displayed in the control is the result of a full AQL query on the Asset Manager database.

### Properties

In addition to the optional properties common to all controls, the "DBQUERYBOX" control recognizes the following properties:

**Physical properties of the "DBQUERYBOX" control**

| Name of the property=Value     | Description of the property                                                                                                | Example                                                                       | Comment |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|---------|
| QUERY="<Full AQL query>"       | Defines the AQL query that returns the information to be displayed in the "DBLIST" control.<br><br>"String" type property. | QUERY="SELECT Name, FirstName FROM amEmplDept WHERE Location=Ariane Building" |         |
| COLTITLE="<Column Column ...>" | Defines the title of columns in the list.<br><br>"String" type                                                             | COLTITLE = "Name FirstName"                                                   |         |

**Physical properties of the "DBQUERYBOX" control, continued**

| Name of the property=Value   | Description of the property                                                                                                                                 | Example            | Comment                                                                                                                                                                                                                                        |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                              | property.                                                                                                                                                   |                    |                                                                                                                                                                                                                                                |
| COLWIDTH="<Width Width ...>" | <p>Defines the size of columns in the list as a percentage of the overall size of the control.</p> <p>"String" type property.</p>                           | COLWIDTH = "50 50" |                                                                                                                                                                                                                                                |
| LISTHEIGHT=<Percentage>      | <p>Defines the size of the "DBQUERYBOX" control proportionally to other "DBQUERYBOX" control in the wizard as a whole.</p> <p>"Long" type property.</p>     | LISTHEIGHT=50      | <p>For example, if there are two "DBQUERYBOX" controls with values "50" and "100" respectively for this property, the second control will be twice as high as the first one. Note that a value smaller than "50" will be replaced by "50".</p> |
| TREE=<TRUE FALSE>            | <p>Displays the data in tree mode (=TRUE) or not (=FALSE).</p> <p>"Boolean" type property.</p>                                                              | TREE=TRUE          | <p>By default, this property is set to "FALSE"</p>                                                                                                                                                                                             |
| MAXSEL = <TRUE FALSE>        | <p>Defines the possibility of selecting more than 99 objects (=TRUE) or not (=FALSE).</p> <p>The selection is limited to 99 objects by default (=TRUE).</p> |                    |                                                                                                                                                                                                                                                |

**Physical properties of the "DBQUERYBOX" control, continued**

| Name of the property=Value | Description of the property                                                                                                                          | Example        | Comment |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|---------|
| MULTISEL=<TRUE FALSE>      | Specifies whether the control supports multiple-selection (=TRUE) or not (=FALSE).<br><br>"Boolean" type property.                                   | MULTISEL=TRUE  |         |
| DBLCLICK=<TRUE FALSE>      | If this property is set to TRUE, Asset Manager will simulate clicking the <b>Next</b> button of the current page.<br><br>"Boolean" type property.    | DBLCLICK=FALSE |         |
| [Value.]ISSELECTION()      | If the selection contains values other than 0 (in other words, the user did not make a selection in the list), this method returns a non-null value. |                |         |

**To modify the maximum number of records returned by the control's query**

By default, the maximum number of returned records is set to **200**.

This prevents the performance of the Web client from being negatively impacted when the number of records that could be returned is high.

If the maximum number of records has been reached, a message is displayed for the user.

To modify the default value:


1. Edit the **aamapi96.ini** file on each of the Asset Manager Web Service servers.
2. In the **[Option]** section, modify or add the following line:

```
/ACWeb/WizQueryBoxMaxLoad=1|<Maximum number of records>
```



## The DBEDIT control

The "DBEDIT" control creates a control identical to that used to populate a field in the Asset Manager database. The control differs according to each field type (date, monetary, and so on).

**Note:** The magnifier button  in this control allows you to select values that are effectively in the database, but you can enter another value.

For this control, the "VALUE" property has the "Variant" (it depends on the control).

### Properties

In addition to the optional properties common to all controls, the "DBEDIT" control recognizes the following properties:

#### Physical properties of the "DBEDIT" control in "Normal" mode

| Name of the property=Value      | Description of the property                                                                                                                                                                                                                                                                                                  | Example                                    |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|
| TABLE="<SQL name of the table>" | SQL name of the starting table.<br>"String" type property.                                                                                                                                                                                                                                                                   | TABLE="amAsset"                            |
| FIELD=<SQL name of the field>   | AQL name of the field used for the control.<br>"String" type property.                                                                                                                                                                                                                                                       | FIELD="seAcquMethod"                       |
| TABLE.LABEL ([iNameType])       | This method returns the name of a data table.<br>The name types are the following: <ul style="list-style-type: none"> <li>• 1 - System name</li> <li>• 2 - SQL name</li> <li>• 3 - Default label</li> <li>• 4 - Description (text in help)</li> </ul> The properties "TABLE" and "NAME" need to have already been populated. | See <a href="#">The DBQUERYBOX control</a> |
| FIELD.LABEL ([iNameType])       | This method returns the name of a given field.<br>The name types are the following: <ul style="list-style-type: none"> <li>• 1 - System name</li> </ul>                                                                                                                                                                      |                                            |

**Physical properties of the "DBEDIT" control in "Normal" mode, continued**

| Name of the property=Value | Description of the property                                                                                                                                                                       | Example |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
|                            | <ul style="list-style-type: none"> <li>• 2 - SQL name</li> <li>• 3 - Default label</li> <li>• 4 - Description (text in help)</li> </ul> <p>The "TABLE" property needs to have been populated.</p> |         |

## The DBTABLE control

The "DBTABLE" control creates a control for entering a table in the Asset Manager database.

This control does not have any additional properties.

## The DBPATH control

The "DBPATH" control creates a control for entering a path of the Asset Manager database.

### Properties

In addition to the optional properties common to all controls, the "DBPATH" control must have the following mandatory property:

**Mandatory logical property of the "DBPATH" control**

| Name of the property=Value      | Description of the property                                                                   | Example                  |
|---------------------------------|-----------------------------------------------------------------------------------------------|--------------------------|
| TABLE="<SQL name of the table>" | <p>Name of the table from which you want to select a path.</p> <p>"String" type property.</p> | TABLE=<br><b>amAsset</b> |


## The LINKEDIT control

The "LINKEDIT" control creates a control for entering a link in the Asset Manager database.

## Properties

In addition to the optional properties common to all controls, the "LINKEDIT" control has the following properties:

### Logical property of the "LINKEDIT" control

| Name of the property=Value            | Description of the property                                                                                                                                                                                                                                                                                                  | Example / Comment                          |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|
| TABLE=<SQL name of the table>         | Name of the table in which you want to select a link.<br><br>"String" type property.                                                                                                                                                                                                                                         | TABLE="amAsset"                            |
| FILTER=<WHERE clause of an AQL query> | Defines an AQL filter.<br><br>"String" type property.                                                                                                                                                                                                                                                                        | This property is optional.                 |
| LINK=<SQL name of the link>"          | SQL name of a link from the table defined in the TABLE property.<br><br>Optional property.                                                                                                                                                                                                                                   | LINK="POrLine"                             |
| ZOOM=<TRUE FALSE>                     | Displays (=TRUE) or not (=FALSE) the magnifying tool.<br><br>This property only applies if the wizard is not modal (MODAL=FALSE property at the root node).                                                                                                                                                                  |                                            |
| SRCCHOICE=<TRUE FALSE>                | Displays (=TRUE) or not (=FALSE) the  icon.<br><br>This property only applies if the wizard is not modal (MODAL=FALSE property at the root node).                                                                                       |                                            |
| TABLE.LABEL([iNameType])              | This method returns the name of the source table of the link.<br><br>The name types are the following: <ul style="list-style-type: none"> <li>• 1 - System name</li> <li>• 2 - SQL name</li> <li>• 3 - Default label</li> <li>• 4 - Description (text in help)</li> </ul> The "TABLE" property needs to have been populated. | See <a href="#">The DBQUERYBOX control</a> |

### Logical property of the "LINKEDIT" control, continued

| Name of the property=Value | Description of the property                                                                                                                                                                                                                                                                                                                | Example / Comment |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| LINK.LABEL([iNameType])    | <p>This method returns the name of the link.</p> <p>The name types are the following:</p> <ul style="list-style-type: none"> <li>• 1 - System name</li> <li>• 2 - SQL name</li> <li>• 3 - Default label</li> <li>• 4 - Description (text in help)</li> </ul> <p>The "TABLE" and "LINK" properties need to have already been populated.</p> |                   |

## The TEXTBOX control

The "TEXTBOX" control creates a control for entering text.

### Properties

In addition to the optional properties common to all controls, the "TEXTBOX" control can have the following property:

#### Physical property of the "TEXTBOX" control

| Name of the property=Value | Description of the property                                                                                                                                                          | Example      |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| MULTILINE=<number>         | This property is set to "0" if the "TEXTBOX" control is mono-line, or a numeric value expressing the percentage of the displayed height of the control if the control is multi-line. | MULTILINE=50 |
| PASSWORD=<TRUE FALSE>      | These properties hide (=TRUE) or display (=FALSE) the typed text.                                                                                                                    |              |

## The CHART control

The "CHART" control enables you to display a graphic. This can be composed of several series.

## Properties

In addition to the optional properties common to all controls, the "CHART" control can have the following properties:

### Logical properties of the "CHART" control

| Name of the property=Value | Description of the property                                                                                                                                                 | Example                            |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| SERIES                     | Defines the list of line names, separated by a vertical bar.<br><br>This list must not be empty, otherwise nothing will appear in the graphic.                              | SERIES="purchase price sale price" |
| VALUES                     | Defines the numeric values of the lines in the graphic.<br><br>Two-dimensional table                                                                                        | VALUES="1 2,1 4"                   |
| FORMAT                     | Defines the data type: <ul style="list-style-type: none"> <li>• Long integer (long)</li> <li>• Double</li> <li>• Number (number)</li> <li>• Percentage (percent)</li> </ul> |                                    |
| SERIE                      | Number of the line you clicked.<br><br>The CHART property must be in interactive mode (=TRUE).                                                                              |                                    |
| INDEX                      | Number of the column you clicked on.<br><br>The CHART property must be in interactive mode (=TRUE).                                                                         |                                    |
| CLICK                      | Calls on this property's script when you click the graphic.                                                                                                                 |                                    |

### Physical properties of the "CHART" control

| Name of the property=Value | Description of the property                                                                                                            | Example |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------|---------|
| MODE                       | Defines the graphic type: <ul style="list-style-type: none"> <li>• MODE=0: vertical bars</li> <li>• MODE=1: horizontal bars</li> </ul> |         |

**Physical properties of the "CHART" control, continued**

| Name of the property=Value | Description of the property                                                                                                                                   | Example                                                              |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
|                            | <ul style="list-style-type: none"> <li>• MODE=2: Segments</li> <li>• MODE=3: Circular (pie chart)</li> </ul>                                                  |                                                                      |
| LABELS                     | Defines the title of columns.                                                                                                                                 | january february                                                     |
| 3D                         | Defines whether the graphic is 3D (=TRUE) or not (=FALSE).                                                                                                    |                                                                      |
| COLORS                     | Defines the list of colors for each line. The objects in the list are separated by a vertical bar.<br><br>RGB value in decimal.                               | 255 16777215 16711680<br><br>Display the colors blue, black and red. |
| INTERACTIVE                | Defines if the graphic is interactive (=TRUE) or not (=FALSE) (in other words, if it's active when the cursor passes over it).                                |                                                                      |
| POPUP                      | Displays (=TRUE) or not (=FALSE) the shortcut menu.                                                                                                           |                                                                      |
| BACKGROUND                 | Defines whether there is a background (=TRUE) or not (=FALSE) for the image.                                                                                  |                                                                      |
| BACKIMAGE                  | Defines the path of the image used as a background for the graphic.<br><br>The "BACKGROUND" property must be validated (=TRUE) to display a background image. |                                                                      |
| STACKED                    | Defines whether the bars of the graphic are stacked (=TRUE) or not (=FALSE).                                                                                  |                                                                      |
| CHARTHEIGHT                | Defines the size of the "CHART" control in relation to other controls in the wizard as a whole.                                                               |                                                                      |
| CAPTION                    | Displays the title.                                                                                                                                           |                                                                      |
| ELEVATION                  | Defines the elevation in degrees of the 3D for pie chart.                                                                                                     |                                                                      |
| ROTATION                   | Defines the angle of rotation for the pie chart.<br><br>Value expressed in degrees.                                                                           |                                                                      |
| DISPLAYLABELS              | Displays (=TRUE) or not (=FALSE) the titles of the columns (LABELS).                                                                                          |                                                                      |
| DISPLAYSLEGEND             | Displays (=TRUE) or not (=FALSE) the key of the lines.                                                                                                        |                                                                      |

## The FILEEDIT control

This control displays a dialog box enabling you to save or load a file.

**Caution:** In the Windows client, you can open a file browser with this control and select the file from either a local or remote directory.

In the web client, this control is displayed in the form of a text box and no icon can be used to start a file browser. Therefore, if this control is used in the web client, the user must manually type the path of the file. In addition, the file must be located on the Web Service server. You can create a shared folder on the Web Service server so that the users can upload files to it.

### Properties

#### Properties of the "FILEEDIT" control

| Name of the property=Value | Description of the property                                                                                                                                                                                                 | Example                       |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| OPENMODE                   | Defines the type of dialog box. <ul style="list-style-type: none"><li>• OPENMODE=1: opens a file.</li><li>• OPENMODE=2: saves a file.</li><li>• OPENMODE=4: opens a folder.</li><li>• OPENMODE=8: saves a folder.</li></ul> |                               |
| FILTERS                    | Defines the criteria for displaying the files listed in the dialog box.                                                                                                                                                     | (* .txt) *.txt (* .scn) *.scn |
| DEFEXT                     | Defines the default file name extension.                                                                                                                                                                                    | (* .scn) *.scn                |

## The TICKEDIT control

This control enables you to insert a timer.

## Properties

### Properties of the TICKEDIT control

| Name of the property=Value | Description of the property                                                                                                     | Example |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------|---------|
| VALUE                      | Representation (in a string form) of the parameters defined by the user in the timer.                                           |         |
| LISTHEIGHT                 | Defines the size of the "TICKEDIT" control in relation to other controls in the wizard as a whole.<br><br>"Long" type property. |         |

## The CALENDAR control

This control enables you to insert a calendar.

## The TIMSPANEDIT control

This control enables you to insert a time-span edit zone.

## The NUMBOX control

This control enables you to insert a numeric type control.

## Properties

### Properties of the NUMBOX control

| Name of the property=Value | Description of the property                                         | Example |
|----------------------------|---------------------------------------------------------------------|---------|
| MINVALUE                   | Defines the minimum value of the number. Infinite value by default. |         |
| MAXVALUE                   | Defines the maximum value of the number. Infinite value by default. |         |



### Properties of the NUMBOX control, continued

| Name of the property=Value | Description of the property                                                                                                                                                                                                                                                      | Example |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| FORMAT                     | <p>Defines the format of the number:</p> <ul style="list-style-type: none"> <li>• LONG in the format defined in the Control Panel of your operating system.</li> <li>• RAWLONG</li> <li>• DOUBLE in the format defined in the Control Panel of your operating system.</li> </ul> |         |

## The COMBOEDIT control

This control enables you to insert a drop-down list type control.

### Properties

#### Properties of the COMBOEDIT control

| Name of the property=Value | Description of the property                                                                                                                                                                    | Example |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| VALUES                     | <p>Defines the value couples "Name" defines the text that is displayed in the control, "Value" the value attributed if this "Name" is selected by the user.</p> <p>"String" type property.</p> |         |

## The DATETIMEEDIT control

The DATETIMEEDIT control.

### Properties

#### Properties of the DATETIMEEDIT control

| Name of the property=Value | Description of the property        | Example    |
|----------------------------|------------------------------------|------------|
| FORMAT                     | Defines the format of the control: | 2002/02/07 |

**Properties of the DATETIMEEDIT control, continued**

| Name of the property=Value | Description of the property                                                                                                                                                         | Example                                        |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
|                            | <ul style="list-style-type: none"> <li>• Date</li> <li>• Time</li> <li>• Datetime</li> </ul> <p>The format of the control depends on the system parameters defines by the user.</p> | <p>13:37:19</p> <p>2002/02/07<br/>13:37:19</p> |

## Using the graphical editor

**Note:** Currently, this functionality is not available in the Web client.

Asset Manager makes it possible for you to create wizards using an integrated graphical editor. This editor aims to simplify and speed up the process of creating a wizard. It is not meant as a substitute for the scripting language, which you must understand to design wizards.

- Overview of the interface
- Creating a new node
- Editing the properties of a node
- Compiling, executing and debugging a wizard

**Note:** To use the graphical editor, the action being created or modified must be a "Wizard" type action (SQL name: seActionType).

This section includes:

|                                                   |     |
|---------------------------------------------------|-----|
| Overview of the interface .....                   | 339 |
| Creating a new node .....                         | 340 |
| Editing the properties of a node .....            | 341 |
| Compiling, executing and debugging a wizard ..... | 341 |

## Overview of the interface

To access the graphical editor, select the **Administration/ Actions** navigation menu item. The graphical editor is shown in the **Wizard** tab of the action detail. It is made up of three parts:





- A toolbar containing the most common functions.
- A **Hierarchy** section that shows a tree-view of the structure of the wizard.
- A section that lists the properties of the node selected in the hierarchy.

### Toolbar

The toolbar allows you to directly activate edit-commands. If you leave the mouse pointer over a ToolTip for a short period of time, a ToolTip is displayed describing the icon.

#### Edit commands

Four edit commands are available:

-  switches the editor between text and graphical mode.
-  moves the node up one level inside its own parent node.
-  moves the node down one level inside its own parent node.
-  deletes the selected node.

#### Execution and debugging commands

These commands allow you to compile, debug and execute the script:

##### Execute and debug button



##### Search tool

The toolbar includes a search tool that can be used to search a character string in the tree-structure of the wizard ("Ctrl+F" puts you directly in this control):

Click this zone and enter the text to search. If successful, Asset Manager moves the selection to the occurrence found ("F3" and "Shift+F3" keyboard shortcuts search the next and previous occurrences respectively).

**Note:** In text mode, the search is full text. In graphic mode, the search only concerns the name of

a property.

## Tree-view of the wizard

The left part of the graphical editor shows a tree-view of the wizard:

When you select a node in the tree, Asset Manager lists the properties associated with this node in the right part of the screen.

## List of properties corresponding to the selected node.

The right part of the screen allows you to enter values for the properties of a node:

Each property has a fixed value or a script. The following color codes are used:

- When a property uses its default value, its name and its value are displayed in gray. You can force another value for this property. It will be displayed in black.
- When a property uses a user-defined value or a script, its name and value are displayed in black.
- When a property is mandatory, its name and value are displayed in red.
- The modified values are displayed in blue.

## Creating a new node

This section details the operations you can perform on a node. The toolbar allows you to move the node up or down or delete it. Here we will deal with the creation of a new node.

**Note:** You can also move a node up or down or delete it using the shortcut menu. In this case, right-click the selected node.

To create node, first select its parent node. For example, to create a new "Page" node, you must first click the "Root" node. When you have selected the parent node, right-click to open the shortcut menu.

The "New" menu item groups together the nodes you can create:

Asset Manager inserts the node in the tree of the wizard.

## Editing the properties of a node

Once the node is created, you can attribute values to the properties of this node. You can do this in the right part of the editor.

The value of a property can be defined in two ways:

- By entering a fixed value
- By defining a script

**Note:** A script always takes precedence over a fixed value. If you assign both a script and a value to a property, Asset Manager will ignore the fixed value and interpret the script.

### Assigning a fixed value to a property


Click the "Value" column of the concerned property directly. According to the type of data accepted by the property (Text, Boolean, Double precision number, and so on), Asset Manager lets you select from a list of possible values or populate a text edit zone.

### Assigning a script to a property

Select the property to which you want to assign a script. The script itself is entered in the **Script** field under the list of properties.

**Note:** By selecting **Restore default value** in the shortcut menu (right-click a property), Asset Manager cancels the fixed value or script and resets the property with its default value. This operation is only possible for properties for which a value or script has been defined by the user (these properties are displayed in black).

## Compiling, executing and debugging a wizard

You can launch the wizard by clicking the  button in the toolbar of the editor. Any errors encountered while executing are displayed in the error history window (accessible via the integrated wizard debugger). By pressing Shift+F9 you can interrupt execution (if the wizard is modal) and activate the debugger.

In this way, you can easily repair and correct errors in the wizard.

**Note:** The execution button is not available if the wizard is contextual.

## Example of creating a wizard

To illustrate the theoretical part of programming a wizard, we will create a "Move" wizard. In association with a "Database" type action, it simplifies the process of moving a user and associated assets from one location to another. The creation of this wizard is described step by step. We invite you to create this wizard by yourself and to consider this section as a guide in case of problems.

|                                                   |     |
|---------------------------------------------------|-----|
| Example of creating a wizard .....                | 342 |
| Step #1- Analyze your needs .....                 | 342 |
| Step #2- Define how the wizard is organized ..... | 344 |

## Example of creating a wizard

The objective of this wizard is move assets from one location to another. For this we need:

- How to identify the assets to be moved

There are three ways to identify the assets to be moved:

- Identifying them according to their user. Once this user is selected, you will need to select the assets to be moved.
- Identifying the assets to be moved directly by selecting their records in the table of assets.
- Identifying the assets to be moved according to their location. You first select a location, then the asset from this location to be moved.

**Note:** We will therefore have to create a user-choice page in order for the user to select a method of selection for the assets to be moved.

- Choosing a new location

To choose a new location for the assets, simply select a record in the table of locations.

## Step #1- Analyze your needs

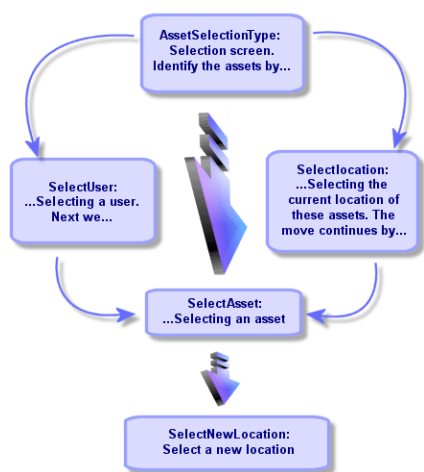
According to the needs defined in step 1, we need to define the organization of the wizard. I.e.:

1. The number of pages.
2. How the different pages are linked.
3. The contents of each of these pages.

**Note:** We know since step 1 that we need to create a selection page. This page will be the first page of the wizard. We will call it "AssetSelectionType".

Now we define how the wizard is organized using the following diagram:

**Wizard - organization example**



Using this flow-chart, we can now define the transitions page by page:

| Page               | Can lead to pages                       |
|--------------------|-----------------------------------------|
| AssetSelectionType | SelectAsset, SelectUser, SelectLocation |
| SelectAsset        | SelectNewLocation                       |
| SelectUser         | SelectAsset                             |
| SelectLocation     | SelectAsset                             |
| SelectNewLocation  | None                                    |

Next we define the contents of the pages. This means the controls that allow the user to make choices:

| Page               | Objective of this page?                               | Control to use     |
|--------------------|-------------------------------------------------------|--------------------|
| AssetSelectionType | Allows the user to choose between three possibilities | A "CHOICE" control |

| Page              | Objective of this page?                                                                                    | Control to use       |
|-------------------|------------------------------------------------------------------------------------------------------------|----------------------|
| SelectAsset       | Allows the user to choose assets from a list of records of the table of assets.                            | An "ADBLIST" control |
| SelectUser        | Allows the user to select a user from the table of departments and employees whose assets are to be moved. | An "ADBLIST" control |
| SelectLocation    | Allows the user to select a current location in the table of locations.                                    | An "ADBLIST" control |
| SelectNewLocation | Allows the user to select a new location in the table of locations.                                        | An "ADBLIST" control |

## Step #2- Define how the wizard is organized

This step consists of writing the script of the wizard. In order to do this, use the descriptions of the structure of each of the nodes of the wizard. The commented source code of the Move wizard is listed below. This code represents but one way of writing the wizard. There are many other ways of writing a wizard performing the same task.

```

;=====
NAME = "Move"
TITLE = "Move user"
VERSION = "699"

;=====
=
;Ask which user to move. By default, use selection in amEmplDept if context is on
this table

;=====
=
{ PAGE pgUser
 TITLE = "Choose the persons who are moving"
 { DBLISTBOX Users
 COLNAME = "Name|FirstName"
 COLWIDTH = "50|50"
 DBLCLICK = 1
 LABEL = "Persons to move"
 MULTISEL = 1
 }
}

```



```
TABLE = "amEmplDept"
{ VALUE =
 if [CurrentTable] = "amEmplDept" then
 RetVal = [CurrentSelection]
 else
 RetVal = ""
 end if
}
VISIBLE = 1
}
{ TRANSITION trPersonToNewLoc
 TO = "pgNewLoc"
}
}

;=====
=
;Ask for new location

;=====
=
{ PAGE pgNewLoc
 TITLE = "Choose the new location"
 { STRING UserName
 VALUE = AmDbGetString("SELECT FirstName + ' ' + Name FROM amEmplDept WHERE
IEmplDeptId IN (" & {pgUser.Users} & ")")
 }
 { LABEL LABEL1
 CAPTION = "Users(s): " & {UserName}
 }
 { DBLISTBOX NewLocId
 COLNAME = "Name"
 COLWIDTH = "100"
 DBLCLICK = 1
 TABLE = "amLocation"
 VALUE = "-1"
 }
 { TRANSITION trNewLocToAssets
 TO = "pgRecap"
 }
}

;=====
=
;Recapitulation
```

```
;=====
=
{ PAGE pgRecap
 TITLE = "Recapitulation"
 { LISTBOX Users
 COLTITLE = "Name"
 COLWIDTH = "100"
 LABEL = "Persons to move"
 MANDATORY = 0
 MULTISEL = 1
 READONLY = 1
 VALUE = ""
 VALUES = AmDbGetList("SELECT FullName FROM amEmplDept WHERE FullName LIKE
LikeParam(amEmplDept_2:FullName)+'%' AND amEmplDept_2:lEmplDeptId IN(" &
{pgUser.Users} & ")","|",",", "=",")
 }
}

;=====
=
;Finish

;=====
=
{ FINISH FINISH
 { DO =
 On Error Goto ErrHandler
 Dim lErr as long

 dim hRecord as Long

 dim iEmplCount as Integer
 iEmplCount = {pgRecap.Users.VALUES.Count()}
 dim iMax as Long
 iMax = iEmplCount

 dim lLocaId as long
 lLocaId = {pgNewLoc.NewLocId}

 lErr = amStartTransaction()

 dim i as Integer
 For i = 1 To iEmplCount
 lErr = AmProgress((100 * i) / iMax)
 lErr = AmLog("Moving the employee" + {pgRecap.Users.VALUES(i,1)})
 hRecord = AmGetRecordFromMainId("amEmplDept", {pgRecap.Users.VALUES(i,0)})
 If hRecord <> 0 then
 lErr = AmSetFieldLongValue(hRecord, "lLocaId", lLocaId)
 End If
 Next i
 }
}

```

```
 lErr = AmUpdateRecord(hRecord)
 lErr = AmReleaseHandle(hRecord)
 End If
Next i

lErr = amCommit()

RetVal = 0
Exit Function

ErrorHandler:
 On Error Goto 0
 AmLog(AmLastError() & " - " & AmLastErrorMsg())
 AmLog("The transaction has been cancelled")
 RetVal = 1
 Exit function
}
SUMMARY = 1
}
```

## Wizard programming case study

### Avoid using local variables to share data

When you program a wizard, we recommend that you do not use local variables to share data. Instead, we recommend the following approaches:

- To share data between different pages, use global variables.
- To pass parameters between wizards in one running chain, use the Params structure:

```
{ Params Params
 { Long lSummary
 Value = 0
 }
}
```

For an example of the Params structure, refer to the "sysFinBudgetClassCreate" or "sysFinBudgetCreate" wizard.

This recommendation is because of a limitation in the Web client. In the Windows client, the dependent properties in the coming pages of a wizard are evaluated. However, in the Web client, these dependent properties are not evaluated.

The following example script helps you understand this limitation:

```
Name = "Wizard"
Version = "10172"
{ Long global_long
 Value = 200
}

{ Page PageA
 { Long A_long
 Value = 100
 }
 { Label A_Label_1
 Caption = {A_long} + 1000
 }
 { Label A_Label_2
 Caption = {global_long} + 1000
 }
 { Label A_Label_3
 Caption = {A_Label_1.Caption} + 10
 }
 { Transition Transition1
 To = "PageB"
 }
}

{ Page PageB
 { OnEnter =
 Dim lErr as long
 lErr = AmSetProperty("PageA.A_long.Value", 300)
 lErr = AmSetProperty("global_long", 400)
 lErr = AmSetProperty("PageC.C_long", 500)
 AmMsgBox({PageA.A_Label_1.Caption})
 AmMsgBox({PageA.A_Label_2.Caption})
 AmMsgBox({PageA.A_Label_3.Caption})
 }
 { Label B_Label_1
 Caption = {PageA.A_long} + 1000
 }
 { Label B_Label_2
 Caption = {PageC.C_Label_1} + 20
 }
 { Transition Transition1
 To = "PageC"
 }
}
```

```
}

{ Page PageC
 { Long C_long
 Value = 0
 }
 { Label C_Label_1
 Caption = {PageC.C_long} + 1000
 }
}

{ Finish Finish
 { Do =
 AmMsgBox({PageB.B_Label_1.Caption})
 }
}
```

The script above defines three long variables:

- A\_long: a variable under the PageA node
- global\_long: a global variable
- C\_long: a variable under the PageC node

The property dependencies are as follows:

| Properties              | Depends on              |
|-------------------------|-------------------------|
| PageA.A_Label_1.Caption | A_long                  |
| PageB.B_Label_1.Caption |                         |
| PageA.A_Label_2.Caption | global_long             |
| PageA.A_Label_3.Caption | PageA.A_Label_1.Caption |
| PageB.B_Label_2.Caption | PageC.C_Label_1.Caption |
| PageC.C_Label_1.Caption | C_long                  |

In PageB, the OnEnter event changes the value of C\_long.

In the Windows client, the following actions occur because of this change:

- The change of C\_long triggers the evaluation of PageC.C\_Label\_1.Caption, which is a dependent property in the coming page (PageC).
- The change of C\_Label\_1.Caption triggers the evaluation of PageB.B\_Label\_2.Caption.

However, in the Web client, because of the limitation, the change of C\_long cannot trigger the evaluation of C\_Label\_1.Caption. Therefore, PageB.B\_Label\_2.Caption is not evaluated.

If you change the scripts in PageB.B\_Label\_2.Caption to:

```
PageC.C_long}+20
```

Then PageB.B\_Label\_2.Caption can be evaluated.

When re-evaluating the properties of previous pages, the Web client behaves the same as the Windows client: the dependent properties in the previous pages are evaluated.

As in the same script above, the OnEnter event in PageB changes the value of A\_long. The following actions then occur in both the Windows and the Web clients:

- The change of A\_long triggers the evaluations of PageA.A\_Label\_1.Caption and PageB.B\_Label.Caption.
- The change of PageA.A\_Label\_1.Caption triggers the evaluation of PageA.A\_Label\_3.Caption.

## Do assign constants to COMOBBOX

We recommend that you assign constants to the values of the COMOBBOX options explicitly.

The following example helps you understand this recommendation.

In the "sysSamEntitlement" wizard, the COMOBBOX control "cbType" in "pgComputers" has scripts bound with its "Value" property:

```
RetVal="No filter," & ComputerType()
```

The "ComputerType()" function is as follows:

```
Function ComputerType() as String
Dim strBuf as String
Dim strTmp as String
Dim strStep as String
Dim strValues as String
Dim strNature(20) As String
Dim i as Long
strBuf = AmGetFieldFormat(AmGetFieldFromName(AmGetTableFromName("amNature"),
"seComputerType"))
strStep=""
i=0
Do While strBuf<>""
i=i+1
strNature(i) = ExtractValue(strBuf, "|")
strTmp = ExtractValue(strBuf, "|")
```

```
 strValues = strValues & strStep & strNature(i)
 if strValues<>"" Then strStep = ","
loop
ComputerType=strValues
End Function
```

As you can find that "seComputerType" is parsed to a string like "xxx|Domain|Virtual Machine|Mobile Device|xxx"; and finally the value of COMOBOX is a string like "No Filter|xxx|Domain|Virtual Machine|Mobile Device|xxx".

Actually, "No Filter" is only a hint text for user and is not one of the options in COMOBOX; other parts of the string are the validate options for users, but do not have corresponding constants.

The constants are very important, because they tell the wizard system which option is selected by user for further processes.

The C++ code automatically generates IDs for the options of COMOBOX, and an ID for "No Filter". However, the hint text should not have a validate ID since it is not a validate option of COMOBOX. Moreover, the auto-generated ID results in duplicate IDs. For example, if the Wizard scripts assign the value of COMOBOX with "No Filter|xxx=1|Domain=2|xxx=3". "No Filter" is then assigned with an auto-generated ID and finally the value is "No Filter=1|xxx=1|Domain=2|xxx=3". In this case, the wizard system cannot identify which option is selected if the user selects 1.

## Do not compare the value of check box control with Boolean

We recommend that you do not compare the value of the CHECKBOX control with Boolean, because the data type of the check box control is integer.

For example, we recommend that you use the script like the following:

```
if {CheckBoxTest.Value} = 0 then AmMsgBox("The checkbox is not selected")
or
if {CheckBoxTest.Value} <> 0 then AmMsgBox("The checkbox is selected")
```

We do NOT recommend that you use the script like the following:

```
if {CheckBoxTest.Value} = 1 then AmMsgBox("CheckBoxTest is selected")
or
if {CheckBoxTest.Value} = True then AmMsgBox("CheckBoxTest is selected")
```

The reason of this recommendation is as follows:

If the integer value of the check box control is 0, the check box is not selected. If the integer value of the check box control is other than 0, the check box is selected.

## The chained wizard will not be started in code line level

The "amWizChain" does not call the changed wizard immediately. The chained wizard only starts after the whole script has been executed.

For example, when you select the wizard of **Asset Lifecycle > Software Asset Management > User Action > Create a new software license**, a wizard chain is started in the "START" node. The scripts in the "START" node are as follows:

```
Dim lErr as Long,hq As long
lErr=amSetProperty("lCntrId",CLng(AmGetUserEnvSessionItem ("SAM",
"SAMCurrentContract")))
If {lCntrId} = 0 Then
 lErr=amWizChain("sysSamGetEnv")
 lErr=amSetProperty("lCntrId",CLng(AmGetUserEnvSessionItem ("SAM",
"SAMCurrentContract")))
End if
if {lCntrId} > 0 Then
 hq=amQueryCreate()
 lErr = AmQueryExec(hq, "SELECT cf_Self, lCostId, lCostCatId FROM amContract WHERE
lCntrId = " & {lCntrId})
 lErr=amSetProperty("strCntrSelf",AmGetFieldStrValue (hq, 0))
 lErr=amSetProperty("pgFinanceInfo.lkeCostCenter",AmGetFieldLongValue(hq, 1))
 lErr=amSetProperty("pgFinanceInfo.lkeCostCategory",AmGetFieldLongValue(hq, 2))
 AmReleaseHandle(hq)

 lErr=amSetProperty("strPossibleModels",amDbGetListEx("SELECT lModelId FROM
amAsset WHERE lLicCntrId=" & {lCntrId} , "", ",", ""))
End if
RetVal="pgSelection"
```

In this example, the chained wizard is actually started after the "START" event.

## The Web client is not compatible with auto-refresh

Do not implicitly refer to the "CAPTION" property.



As an example, in the "pgFinanceInfo" page of the **Create a new software license** ("sysSamCreateLicpfi") wizard, the object "tr" of "TRANSITION" has a condition, which contains the following scripts:

```
IF {lkeMaintContract.Value} > 0 AND ({lkeModel} > 0 OR {dtStart} <> "" OR {dtEnd} <> "" OR {numCost} > 0) THEN
 RetVal = AmSetProperty("pgFinanceInfo.lbError.Caption", "<P><HR><CENTER>" & FormatResString("Please either select an existing contract $1or$2 create a new one", "<U>", "</U>") & "<HR>")
 AmMsgBox(FormatResString("Please either select an existing contract $1or$2 create a new one", "", ""))
 RetVal = FALSE
ELSE
 RetVal = AmSetProperty("pgFinanceInfo.lbError", "")
 RetVal = TRUE
END IF
```

The following script does not cause any warning or error in runtime; however, the script is illegal in C++:

```
RetVal = AmSetProperty("pgFinanceInfo.lbError", "")
```

The purpose of the scripts is to set pgFinanceInfo.lbError.caption with a NULL string. What C++ does is the following:

1. Create and initialize a string for "pgFinanceInfo.lbError.Caption" with a NULL string.
2. Try to find the "pgFinanceInfo.lbError" property and set the NULL string to it.

In the second step, C++ cannot find any property called "pgFinanceInfo.lbError", and hence is about to throw an error. However, the code finds the new value is NULL and the old value (initial value) is also NULL; so C++ does not throw the error.

If you change the script to:

```
RetVal = AmSetProperty("pgFinanceInfo.lbError", "text")
```

You receive the error that resembles the following:

```
Error (12,001): Property 'lbError' unknown in object 'pgFinanceInfo' of the wizard. ('Line 6 of script 'pgFinanceInfo.tr.Condition, line 389')
```

```
Unable to complete operation in current state. ('Line 6 of script 'pgFinanceInfo.tr.Condition, line 389')
```

To avoid this error, change the script to the following to refer to the "CAPTION" property explicitly:

```
RetVal = AmSetProperty("pgFinanceInfo.lbError.caption", "text")
```

**Note:** Although it is illegal for wizard to refer to the CAPTION property implicitly, it is acceptable to refer to the "VALUE" property in controls (for example, "DBLISTBOX") implicitly.

## Do not apply dual-source for the VALUE property

Do not apply multiple sources for the VALUE property. Applying multiple sources for the VALUE property may cause the following issues:

- If the VALUE property can be affected in multiple places, the order to affect the property is not predictable.
- Auto-refresh on windows client will not work on web-based wizard.

## Frequently asked questions

This chapter aims to answer the questions you are likely to ask when creating a wizard.

|                                                                                                         |     |
|---------------------------------------------------------------------------------------------------------|-----|
| {IbxMyListBox.Values.Count} does not work .....                                                         | 354 |
| {IbxMyListBox.Line(IRow)} does not work .....                                                           | 355 |
| {IbxMyListBox.Values.Line({IbxTmp})} does not work .....                                                | 355 |
| Assigning fixed value to a property does not work .....                                                 | 355 |
| Executing a wizard which creates an asset in the database causes error .....                            | 356 |
| Incomplete error message appears when executing wizard .....                                            | 356 |
| Performance is impacted when the "DBLISTBOX" control is used in a wizard page .....                     | 356 |
| How to allow or forbid editing certain columns in the "LISTBOX" control? .....                          | 357 |
| How to handle error "Component ID AcwOFFScreenFilter has already been found in the view" ..             | 357 |
| What do I need to do to make a wizard open a detail screen? .....                                       | 358 |
| What is the difference between the "COLNAME" and "COLTITLE" properties of a "LISTBOX"<br>control? ..... | 358 |
| The word "query" cannot be used in any wizard elements .....                                            | 359 |

## {IbxMyListBox.Values.Count} does not work

### Answer

You must enter opening and closing parentheses in the syntax of the method. Here is the corrected code:

```
{lBxMyListBox.Values.Count()}
```

## {lBxMyListBox.Line(lRow)} does not work

### Answer

The "LINE" method is associated with the "VALUES" property of the "LISTBOX" control. Here is the corrected code:

```
{lBxMyListBox.Values.Line(lRow)}
```

## {lBxMyListBox.Values.Line({lBxTmp})} does not work

### Answer

You cannot use a referenced property in a method. Here is what you should write:

```
Dim lRow As Long
lRow = {lBxTmp}
{lBxMyListBox.Values.Line(lRow)}
```

## Assigning fixed value to a property does not work

For example:

```
{Property} = 123
```

### Answer

To assign a value to a property, you need to use the "AmSetProperty()" function, as shown below:

```
Dim irc as Integer
irc= AmSetProperty("Property", 123)
```

**Note:** Don't forget to recover the return code ("irc" in this example), even if you are don't need to use it.

## Executing a wizard which creates an asset in the database causes error

When executing a wizard that creates an asset in the database, the following error message appears:

```
12001 - You do not have write-access rights
```

This message appears even if the user is connected as administrator.

### **Answer**

This message appears when a write-access is attempted outside of the wizard "FINISH.DO" node. The wizard does the following:

1. Collects information via a series of pages (write-access forbidden even for the Asset Manager administrator)
2. Executes the script contained in the "FINISH.DO" node (write-access authorized according to the user's access rights)

## Incomplete error message appears when executing wizard

### **Answer**

Press SHIFT+F9 to display the debugger. Error messages in the history window are often more explicit.

## Performance is impacted when the "DBLISTBOX" control is used in a wizard page

### **Answer**

This problem occurs when you use the "DBLISTBOX" control in conjunction with a filter. When this is the case, each time the selection changes, a query is sent to the database to make sure that the

selection corresponds to the filter. This additional query is not performed when the selection is set by the user.

## How to allow or forbid editing certain columns in the "LISTBOX" control?

### Answer

Use the "EDITABLE" property of this control. The value assigned to this property is a string made up of "0" and characters separated by the pipe character "|", which is used as a column separator. "0" defines the column as "non-editable", "1" as "editable". If you omit a value, the corresponding column will not be editable, only columns 2 and 4 are editable:

```
EDITABLE = "|1||1"
```

## How to handle error "Component ID AcwOFFScreenFilter has already been found in the view"

You use Websphere as the application server. You modify a wizard and then log on to the web client. In this scenario, you cannot run any wizard without restarting the web service. If any wizard is triggered on the web client, the "Component ID AcwOFFScreenFilter has already been found in the view" error occurs.

### Answer

1. Go to the webtier folder \WEB-INF\classes.
2. Copy the following two classes to websphere SharedLibrary.
  - com\sun\facelets\impl\DefaultFaceletFactory.class
  - com\hp\ov\ac\web\rendering\facelet\LruCache.class

After you copy these two classes, there will be three jsf jars and one com package which has the two classes above.

## What do I need to do to make a wizard open a detail screen?

### Answer

You need to use DDE calls (via a function) inside the wizard. The wizard must not be modal. Here is an example of how to open the table of asset from inside a wizard:

```
Dim irc as Long
irc = AmActionDDE("am", "Asset Manager", "OpenTable(amAsset)")
```

## What is the difference between the "COLNAME" and "COLTITLE" properties of a "LISTBOX" control?

### Answer

The titles of columns in a "LISTBOX" control can be defined automatically or manually:

- The "COLNAME" property, associated with the "TABLE" property allows you to automatically define the titles of columns in a "LISTBOX" control using field labels from the database.
- The "COLTITLE" property if it is populated forces the titles of the columns. If this property is not defined, the column titles will be those defined by the "COLNAME" property.

Thus the following example:

```
...
TABLE = "amEmplDept"
COLNAME = "Name||FirstName"
COLTITLE = "|A|B"
...
```

displays the following labels in the columns of the "LISTBOX" control: Name, A, B.

The "COLNAME" property also defines the type of control used when the column values are editable.

## The word "query" cannot be used in any wizard elements


### **Answer**

An unexpected error occurs if using the word "query" in a wizard element. Because the word "query" is a reserved keyword in wizard scripts, using it in any wizard elements such as the wizard title will result in an error.

# Chapter 15: News

This chapter explains how to broadcast and manage news with Asset Manager.

You access the list of news using the **Tools/ News** menu.

You activate/deactivate the news marquee using the **Windows/ Display news marquee** menu or the  in the toolbar.

This chapter includes:

|                                 |     |
|---------------------------------|-----|
| Definition of a news item ..... | 360 |
| Overview of news .....          | 360 |
| Importance of news items .....  | 361 |
| Message to broadcast .....      | 361 |
| News broadcast list .....       | 361 |
| Display the news .....          | 362 |

## Definition of a news item

A news item is a piece of information that you want to broadcast to a designated group of persons for a specified period of time.

These persons belong to employee groups.

In general, this type of information is of a short-lived nature.

Example of a news item: "The XXX server will be down for maintenance between 11:00 and 12:00 on February 10, 2011".

## Overview of news

This section includes:

|                            |     |
|----------------------------|-----|
| Creating a news item ..... | 361 |
| Reading news .....         | 361 |



## Creating a news item

Users with rights in the News table can create news using the **Tools/ News** menu.

In the news detail you will see the:

- Message
- Broadcast list of the message.
- Validity period of the message.

## Reading news

The news marquee enables any user to read the messages broadcast to the group of which they are a part.

## Importance of news items

To define the importance of a message, you must populate the **Importance** field (SQL: seSeverity) in the news detail.

Each level of importance is associated with a color that you choose in the **Color** field (SQL: IColor), which then appears in the news marquee.

## Message to broadcast

The **Message** tab of a news detail contains the text to broadcast.

A message can contain 255 characters.

## News broadcast list

The **Broadcast** tab of a news detail lists the employee groups who can read the news item.

**Note:** Any helpdesk administrator, employee group manager and Asset Manager administrator will see a news item provided they are registered in the news broadcast list.

This section includes:

|                                                    |     |
|----------------------------------------------------|-----|
| All employee groups option (SQL: bAllGroups) ..... | 362 |
| Include sub-groups option (SLQ bChildGrps) .....   | 362 |

## All employee groups option (SQL: bAllGroups)

If you select this option, the members of all employee groups will see the message.

Otherwise, only the members of the selected employee groups (in the **Broadcast** tab's list) will see the message.

Use the ,  and  to add, delete, view or modify the employee groups in the broadcast list.

## Include sub-groups option (SLQ bChildGrps)

A message can be broadcast to all employee groups or a selection of groups. If you select this option (which is selected by default) the set of sub-groups of the selected groups will see the message. This is because the employee groups are arranged hierarchically.

## Display the news

To view the news:

1. Activate the news marquee.
2. Use the buttons in the news marquee to scroll through the messages.


See ["Activating the news marquee" on the next page.](#)

You can define:





- The colors of how messages are displayed according to their importance.
- The automatic-refresh mode of the news marquee.

## Activating the news marquee

Any Asset Manager user can activate the news marquee. There, users will see those messages broadcast to the employee group to which they belong. The news marquee can be activated/deactivated in two ways:

- Using the **Window/Display news marquee** menu.
- Using the  icon in the toolbar.

### News marquee buttons

|                                                                                   |                                                                                                                                                               |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Click this button to read new messages before waiting for the news marquee to check for them at its next defined interval (via the <b>Edit/Options</b> menu). |
|  | Click this button to display the previous message.                                                                                                            |
|  | Click this button to display the next message.                                                                                                                |
|  | Click this button to interrupt or continue the scrolling of the news in the marquee.                                                                          |

## Data health check



The health check process helps you monitor the data integrity. To ensure the accuracy of the license calculation, you need to check the inventory data and counter settings using the health check process.

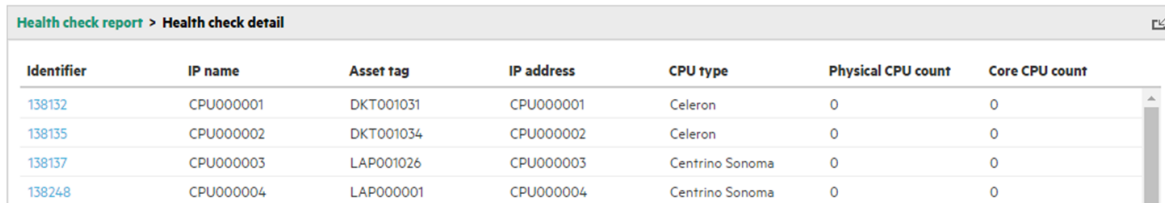
**Note:** This process is only available on the web client.

## Health check widget

You can view the health check result by adding a widget named “Health check report” to the home page, as shown in the following figure.

| Check item                 | Check rule                                                                    | Unhealthy records | Rule identifier | Context          |
|----------------------------|-------------------------------------------------------------------------------|-------------------|-----------------|------------------|
| <b>Health check report</b> |                                                                               |                   |                 |                  |
| Counter common check       |                                                                               |                   |                 |                  |
|                            | Counters (created by template) without link to license or installation models | 0                 | 131961          | amSoftLicCounter |
|                            | Counters whose installation models are not used by any software installation  | 0                 | 131962          | amSoftLicCounter |
|                            | Counters without brand or company information                                 | 3                 | 131960          | amSoftLicCounter |
| Inventory data             |                                                                               |                   |                 |                  |
|                            | Computers with incomplete CPU information                                     | 333               | 131953          | amComputer       |
|                            | Software installations missing computer information                           | 31                | 131959          | amSoftInstall    |
|                            | Virtual machines without a host relationship                                  | 124               | 131952          | amComputer       |
|                            | Virtual machines without virtualization technology                            | 93                | 131956          | amComputer       |
| LPAR inventory data        |                                                                               |                   |                 |                  |
|                            | LPARs with incorrect LPAR information                                         | 0                 | 131954          | amComputer       |
|                            | Shared LPARs with incorrect capacity information                              | 20                | 131955          | amComputer       |
| Utilization                |                                                                               |                   |                 |                  |
|                            | Computers not scanned in the past 90 days                                     | 64                | 131957          | amComputer       |
|                            | Software not used in the past 90 days                                         | 823               | 131958          | amSoftInstall    |

You can click a check rule to drill down into a detailed widget, as shown in the following figure.



| Identifier | IP name   | Asset tag | IP address | CPU type        | Physical CPU count | Core CPU count |
|------------|-----------|-----------|------------|-----------------|--------------------|----------------|
| 138132     | CPU000001 | DKT001031 | CPU000001  | Celeron         | 0                  | 0              |
| 138135     | CPU000002 | DKT001034 | CPU000002  | Celeron         | 0                  | 0              |
| 138137     | CPU000003 | LAP001026 | CPU000003  | Centrino Sonoma | 0                  | 0              |
| 138248     | CPU000004 | LAP000001 | CPU000004  | Centrino Sonoma | 0                  | 0              |

By default, the detailed widget only shows the first 100 records. If you want to change this behavior, configure the **MaxRecords** parameter in the **Health check report** widget (the first widget). This setting applies to all health check rules. The range is 1 to 500 due to performance considerations.



## Managing health check rules

Besides out-of-the-box health check rules, you can add new check rules. To do this, open **Asset lifecycle** -> **Software asset management** -> **Manage health check item** -> **Edit check items and check rules**.

| <input type="checkbox"/> | Name                  | Category              | Show in widget |
|--------------------------|-----------------------|-----------------------|----------------|
| <input type="checkbox"/> | Utilization           | Utilization           | Yes            |
| <input type="checkbox"/> | Counter common check  | Software counter      | Yes            |
| <input type="checkbox"/> | <b>Inventory data</b> | <b>Inventory data</b> | <b>Yes</b>     |
| <input type="checkbox"/> | LPAR inventory data   | Inventory data        | Yes            |

Display selections 0 selected records  Action on the selections in current page Page 1 of 1 Total records: 4 20 records per page

**Detail of health check item 'Inventory data'**

Name: Inventory data

Description: Check hardware and software installation integrity.

Category: Inventory data  Show in widget \* Code: HCI\_INVENTORY

**Check rules:**

| <input type="checkbox"/> | Name                                                | Threshold |
|--------------------------|-----------------------------------------------------|-----------|
| <input type="checkbox"/> | Virtual machines without a host relationship        | 0         |
| <input type="checkbox"/> | Computers with incomplete CPU information           | 0         |
| <input type="checkbox"/> | Virtual machines without virtualization technology  | 0         |
| <input type="checkbox"/> | Software installations missing computer information | 0         |

When the **Show in widget** check item option is selected, all check rule that belongs to the check item will be shown in the widget. Otherwise, the widget will ignore the check item. There are several check rules under the check item. By clicking a check rule, you will see the definition of the check rule, as shown in the following figure.

Welcome to HPE Asset Manager > Detail of health check rule 'Virtual machines without a host relationship'

Name: Virtual machines without a host relationship \* Code: HCR\_CPU\_VMNOHOST

Description: Look for virtual machines which are missing the relationship with host.

**AQL Query**

Columns: TcpIpHostName,AssetTag,TcpIpAddress

Condition: Virtual machines without a host link

Threshold: 0

- The **Threshold** field of the check rule is used to control whether the check rule will be shown in the widget. If the number of matched records is greater than or equal to the value of the threshold, the result will be shown in the widget. Otherwise, the result will be ignored by the widget.

- The **Columns** field allows you to specify the fields you want to show in the detailed report. The context table is defined in **Condition**, which is an AQL query.
- The **Validate rule syntax** action is used to check the grammar of the rule.

# Send documentation feedback

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

## **Feedback on Advanced Use (Asset Manager 9.61)**

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to [ovdoc-ITSM@hpe.com](mailto:ovdoc-ITSM@hpe.com).

We appreciate your feedback!