Peregrine

AssetCenter

Administration

Peregrine SYSTEMS

PART NUMBER ACT-4.0.0-ENG-01008-00064

This edition applies to version 4.0.0 of the licensed program

AssetCenter

Peregrine Systems, Inc.
Worldwide Corporate Campus and Executive Briefing Center
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Table of Contents

Chapter 1. Database options 27

Conventions 27

Chapter 2. Structure of the AssetCenter database 33

Foreword 33
Definition of a database 35
Structure of the Database.txt and Tables.txt files 35
Description of the tables 37
  Identifying AssetCenter tables 37
  Table description strings 37
Description of fields 38
  Identifying AssetCenter fields 38
  Field types and data entry formats 39
  dtLastModif fields 43
  "FullName" fields 43
Description of links 44
  Identifying links 44
  Type of a link 45
  Cardinality of a link 46
  Intermediary tables 47
  Contextual links 49

Description of indexes 49
Identifying indexes 49
Uniqueness of indexed field values 50
Customizing the database 50
Customizing tables 51
Customizing fields and links 51
Customizing the indexes 52
Default values for fields and links 53
Counters in field default values 54
Errors in the description of default values 54
HTML tags recognized in help on fields and links 55

Chapter 3. Creating an AssetCenter database 57

Declaring a connection to an empty shell 58
Entering the authorization string 58
Entering the administrator password and "physically" creating the database 60
Modifying the authorization string 61
Modifying the authorization string using AssetCenter Database Administrator 61

Chapter 4. Managing itemized lists 63

Free itemized lists 63
Values of itemized lists 64
Open itemized lists 65
Closed itemized lists 66
System itemized lists 66
Values in system itemized lists 66

Chapter 5. How the history function works 69

Record creation 71
Modification of a field in the table or a 1 link (for example: User of an asset) 71
Addition of a n link to another table (for example: Assets covered by a contract) 71
Deletion of a n link to another table 72
Modification of a n link to another table 72
Keeping history of features associated with records 72
Adding a feature 72
Deleting a feature 73
Modifying the value of a feature 73
Creating, deleting or modifying history lines 74

Creating history lines 74

Chapter 6. Managing user access to AssetCenter databases 75

Importance and overview of database access rights 76
Defining access conditions for each user 76
Managing user connections 77
How to ensure data security and confidentiality 77
Definitions 78
Definition of a user profile 78
Definition of a user right 78
Definition of a functional right 78
Definition of an access restriction 79
Defining access conditions 79
Defining user profiles 79
Defining user rights 79
Defining functional rights. 82
Defining access restrictions 82
Defining AssetCenter users 84
Defining a database administrator 85
Defining a non-administrative user 85
Managing user connections 85
Database access types 85
How connection slots work 87
Destroying connection slots 88
Managing passwords 90
Password for the Admin login 90
Modifying the password for the Admin login 91
User password 91
Lost passwords 91

Chapter 7. Writing queries in AQL 93

Presentation 93
Queries in AssetCenter 93
AQL 94
Recommendations for writing AQL queries 97
Presentation of AQL joins 98
Reason for and usefulness of primary key 0 records 99
Usage of NULL 101
Self 102
CurrentUser 102
System itemized-lists 103
Hierarchic tables 104
Simplified AQL notation 105

Sorts and indexes 107
- Example 107
- How to force the indexes 108
- Sort order 108
- Precautions 109

The query editor 110
- Overview 110
- Accessing the query editor 111
- Creating a query using the query editor 112
- Fields used in queries 114
- Writing expressions 114
- Constants 115

AQL syntax 116
- Conventions 117
- Syntax of queries 117
- FROM clause- Tables involved in a query 118
- Elements of a query 120
- WHERE clause 125
- GROUP BY clause 126
- HAVING clause 127
- ORDER BY clause 128

AQL function reference 128
- Aggregate-type AQL functions 129
- String-type AQL functions 130
- Date-type AQL functions 131
- Numeric-type AQL functions 134
- Test type AQL functions 134

Examples of queries 134
- To compare a field in the main table with a value 135
- To compare a link in the main table with another link 135
- To compare a link in the main table with a value 135
- To compare according to a field in a table linked to the main table . . . 136
- Hierarchic tables 136
- Query combining two conditions 136
- Comparison of a field with numbers, dates or text 137
- Query concerning a feature 137
- To search records according to an expression 137
- To search a field that is not populated 137
- To search for the absence of a link 138
- Query with alias 139

Chapter 8. Editing forms 141

Definition of a form 141

Creating forms 141
  Basic information 141
Editing forms and objects 142
  Fixed text 143
  Formulas 144
  Lists 144
  Images 144
Properties of object in forms 145
  Position and size 145
  Properties 145
Design grid 147
Form page setup 148
How to easily create regular reports 149

Chapter 9. Editing reports 151

Operation and installation of the report generator 151
  Overview 151
  Installing, configuring and inserting Crystal reports in your database . . 152
Detail of a report 152
  File 152
How to modify a Seagate Crystal report 153
Crystal Reports statistics 153
  Nature 154
  button 154
  button 154
Creating a detail report 154
  Example of utilization 154
  Configuring reports under Crystal Reports 154

Chapter 10. Read-only access to the database using the AssetCenter ODBC driver 157

Overview of access to the AssetCenter database 157
  Installing the ODBC driver 157
  When to use the ODBC driver 158
  Data items accessible via the ODBC driver 159
  Which ODBC connection to use 160
Example: Creating a report under Crystal Reports with the ODBC driver . . 160

Chapter 11. Defining actions 161

Definition of an action 161
  Functional domain 162
Creating an action 163

Types of actions . . . 163
General method . . . 166
Populating the DDE tab . . . 168
Populating the Messaging tab . . . 169
Examples of actions . . . 172
Example of an executable-type action. . . . 172
Example of a DDE-type action . . . 173
Example of Messaging type action . . . 178
Example of a Script type action . . . 179
Using variables . . . 181
Specificities of the Sybase SQL Anywhere database engine . . . 181
Testing an action . . . 182
Calculate button . . . 182
Execute button . . . 182
Executing an action . . . 182
Multiple-selection in lists . . . 183
Wizard-type actions . . . 183

Chapter 12. Managing deadlines with AssetCenter Server . . . 185

Overview of AssetCenter Server . . . 185
Executing AssetCenter Server . . . 187
Recommendations . . . 187
Launching AssetCenter Server . . . 187
Executing AssetCenter Server manually under Windows . . . 188
Using the security-integrated Windows NT . . . 189
If you exit AssetCenter Server . . . 195
AssetCenter Server and the messaging system . . . 195
Main screen of AssetCenter Server . . . 196
General options of AssetCenter Server . . . 197
Database reconnection delay . . . 197
Messaging reconnection delay . . . 197
Log file . . . 197
Time zones . . . 198
Configuring the modules monitored by AssetCenter Server . . . 199
Introduction . . . 199
Verification schedules . . . 200
Verify history lines module (History) . . . 203
Verify stocks module (Stock) . . . 203
Verify alarms module (Alarms) . . . 204
Calculate rents module (Rent) . . . 206
Calculate stipulated loss values module (LostVal) . . . 208
Split expense lines in cost centers module (CostCenter) . . . 208

Signal presence to database server module (UpdateToken) 210
Search for new workflow execution groups module (WorkflowFinder) 211
Execute workflow rules for execution group 'G' modules 211
Add the computers listed in the NT domain to the database module (AddCpu) 213
Add NT users to the database module (AddUser) 216
Create assets, consumptions, etc. corresponding to items received module (Delivery) 219
Send the scanner to the computers module (SendScan) 219
Get the results of the scanners module (GetFsf) 224
Update the database with the results of the scanners module (IddAc) . 228
Verify null-identifier records module (History) 231
Purge the input-events table module (PurgeEventInTable) 231
Purge the outgoing-events table module (PurgeEventOutTable) 231
Update statistics for tables module (Stats) 232
Activating the verification immediately 232
Administrating AssetCenter Server by the Web 232
Launching AssetCenter Server as an NT service 233
Accessing the AssetCenter Server service via the Web 234
Driving the AssetCenter Server service via the Web 235

Chapter 13. Managing messages 239

Overview of messaging 239
Overview 240
How to issue messages 240
How to consult messages 241
Acknowledgement of receipt 241
Configuring AssetCenter to use messaging systems 241
Multiple recipients 242
SMTP protocol 242
MAPI protocol 245
VIM protocol 247
AM protocol 251
Common connection problems 253
Test to validate a connection to a messaging system 253
Test to perform if a problem is encountered 254
Troubleshooting 254

Chapter 14. Workflow 257

Definitions 257
Workflow 258

Workflow activity 259
Workflow event 259
Workflow transition 259
Workflow task 260
Workflow activity assignee 260
Definition of a workflow execution group 260
General overview 260
Main tables involved in workflow 262
Using the graphical workflow editor 264
Activities 264
Events 265
Transitions 266
Other functionality 266
How to implement workflow 267
Example of workflow used in request approval 267
Aim 267
Preliminary steps 269
Creating activities 273
Creating the start event 277
Creating transitions 278
Example of activating a workflow instance 279
The context of a workflow instance 283
Defining the context of a workflow instance 284
Object referenced by a workflow instance 284
Limiting workflow instances in progress for a given object 284
Workflow roles 286
Workflow role type 286
Defining the assignee of an activity 288
Activities 288
Question type activities 288
User-action type activity 290
Automatic-action type activity 291
Test/ Script type activities 292
Start activity 293
Activity templates 293
Triggering activities 294
Tasks 294
Creating tasks 295
Automatic action or test/ script type activity 295
Displaying the list of tasks in progress 296
Performing a user task 296
Assigning user tasks 297
Administrating a workflow task 297
Events 298

System events 298
Alarm events 299
User events 299
General conditions of activation 302
Processing of events 303
Application: Implementing a synchronous workflow scheme 308
Terminal event 309
Workflow transitions 310
Workflow alarms and time limits 310
Time limit 311
Workflow alarms 312
Workflow execution groups 313
Workflow tracking 313

Chapter 15. Importing data 315

Overview of importing data 316
A single text file 316
Several text files 316
A complete database 316
Recommendations 317
Default values for mandatory fields 317
Avoid using "Id" field as reconciliation keys 318
Backup the AssetCenter database 318
Avoid simultaneous access to the AssetCenter database 318
Constraints to be respected when importing data into a field 318
According to the value of the "UserType" property of the target field . . 320
According to the value of the "type" property of the target field 322
According to the value of other properties of the target field 322
Importing text files or an ODBC database 323
Before importing text files 323
Step 1: Select the text files or ODBC database to be imported 324
Step 2: Define the parsing of files or tables to be imported 325
Step 3: Describe the fields to be imported 330
Step 4: Map source fields to target fields in the AssetCenter database . . 333
Step 5: Map each text file or source table to a target table 334
Step 6: Map the fields to be imported to fields in the AssetCenter database 334
Step 7: Add additional calculated fields in the source file 340
Step 8: Special cases 344
Step 9: Examples of using keys 345
Step 10: Configuring the transfer 351
Step 11: Data transfer 352
Importing an AssetCenter database 353

Saving and executing an import script 354
To save a script 354
To modify a script 355
To create a new script when you are defining import settings 355
To execute a script 356
Executing an import script from a command prompt 356
How it works 356
Syntax 356

Chapter 16. Exporting data and managing SQL views 359

Definitions 359
Export scripts 359
Export queries 360
Exporting data from the AssetCenter database 360
Exporting data using an export script 361
Exporting data using the File/ Export list menu item 361
Managing SQL views in the AssetCenter database 362
Recommendations 363
Defining an export script 363
Methodology 363
Defining export queries 364
Output format of an export script 367
Actions concerning SQL views 368
Executing an export script 369
Executing an export script from AssetCenter Export 369
Executing an export script from DOS 370

Chapter 17. Using scripts 373

Definition of a script 373
Overview 373
Information about this version of Basic 374
Data access notation 375
Applications of scripts 375
Introduction to functions 376
Definition of a function 377
Built-in functions and programmable functions 377
Function and parameter types 379
Classifying Basic functions 381
First steps in writing scripts 381
Example scenario 382
Step 1: Create the feature "Tutorial" 382
Step 2: Open the edit window 383
Step 3: Analyze and define the algorithm 383

Step 4: Draw up the Basic script 383
Step 5: Test the Basic script 384
Script library 385
Concepts 385
Creating a script library 385
Calling up a script in a script library 386
Tips and warnings 386
Precautions when using programmable functions 387
Format for "Date+Time" constants in scripts 387
Format for "Duration" constants in scripts 388
Read and write access to a system itemized list 388
The "CurrentUser" virtual link 390
Commenting a Basic script 390
Triggering an error message 391
First example 392
Description of the problem 392
Step 1: Analyze and define the algorithm 392
Step 2: Draw up the Basic script 393
Step 3: Test the Basic script 394
Second example 394
Description of the problem 395
Step 1: Analyze and define the algorithm 395
Step 2: Draw up the Basic script 395
Step 3: Test the Basic script 396

Chapter 18. Managing calendars 397

Overview of calendars 397
The calendar detail 397
Using calendars 398
Impact of calendars on certain areas of functionality 398
Methodology used to create a calendar 398
Description of how to create a calendar 399
Entering general information 399
Populating the Timetable tab 399
Populating the Exceptions tab 401
Checking the calendar 404

Chapter 19. Managing time zones 407

Why manage time zones? 407
Example 408
Implementing time zones 408
Creating time zones 409
Managing a time zone 409

Format of Daylight time field 410
Values of the <Year> argument 410
Values of the <DaylightInfo> argument 411
Example 415
Managing time zones in AssetCenter Server 418
Tests to perform 418
Frequency of the test 419
Consequences for various operations 420
Creating the database 420
Connecting to a database 421
Import and Export 423

Chapter 20. Using AssetCenter as a DDE server 425

Definition of a DDE server 425
DDE call mechanisms 425
DDE service 426
DDE topic 426
DDE command 426
Introduction to DDE commands 427
Steps to follow 427
Particularities 427
Global commands 428
Connect(<Cnx>, <User>, <Password>) 428
Disconnect() 429
ExecuteAction(<ActionName>) 430
ListAllTables([Mask]) 430
ListOpenTables([Mask]) 431
OpenView(<ViewName>) 432
Commands associated with a table 433
OpenTable(<Table>) 433
CloseTable(<Table>) 434
<Table>.GetRecordCount() 434
<Table>.SetViewMode(<Mode>) 435
<Table>.SetRecordMode(<Mode>) 435
<Table>.ListAllFields([Mask]) 437
<Table>.ListAllLinks([Mask]) 437
<Table>.SetFilter(<Condition>) 438
<Table>.SetSelection(<Condition>) 439
<Table>.GetSelectionId() 440
Commands associated with a table and a field or a link 440
<Table>:<Object>.AddLink() 441
<Table>:<Object>.GetValue() 441
<Table>:<Object>.Hilight() 442

<Table>:<Object>.RemoveLink() 443
<Table>:<Object>.SetFocus() 443
<Table>:<Object>.SetValue(<Value>) 444
<Table>:<Link>.SetValueWhere(<Condition>) 445
<Table>:<Object>.Show() 446
Introduction to Examples of DDE calls 446
First scenario: AssetCenter internal DDE calls 447
Answering the call 447
Entering data 448
Creating the feature 449
Second scenario: DDE calls from Excel 449
Description of the macro 449
Source code of macro 450
Third scenario: DDE calls from Visual Basic 451
Source code of the program 451
Before starting 451
Executing the program 452

Chapter 21. Calculated fields 455

Definition of a calculated field 455
Usefulness of calculated fields 456
Creating a calculated field 456
Introduction 457
Methodology used to create calculated fields 459
Using calculated fields 462
Using a calculated field in the configuration of a list 462
Filtering the records of a table 463
Referencing a calculated field 463

Chapter 22. Introduction to wizards 465

Definition of a wizard 465
Who are wizards aimed at? 466
Classification of wizards 466
Wizards that exchange information with the AssetCenter database . . . 466
Independent wizards 467

Chapter 23. Creating wizards 469

Conventions used 469
Definitions 470
Twip 470
Control 470
Node 471

- Object . . . 471
- Parent object and child object . . . 471
- Full name of an object . . . 471
- Variable . . . 472
- Transition . . . 472

Structural model . . . 473

Model of a wizard page . . . 476

General points . . . 476

Generic structure and syntax . . . 477

Properties of a node . . . 478

- Declarative model . . . 478
- Defining a constant as a value for a property . . . 479
- Referencing a property . . . 479
- Defining a script as value for a property . . . 480
- Methods applicable in properties . . . 481
- Table type property . . . 482
- Using the global variables CTXTTABLE and CTXTSELECTION . . . 483

Sequencing wizards . . . 484

- Execution . . . 484
- Parameters . . . 484

Basic functions . . . 485

Definition of a Root node . . . 485

Syntax of a Root node . . . 485

Properties of a Root node . . . 486

Sub-nodes of a Root node . . . 490

Definition of a Page node . . . 491

Syntax of a Page node . . . 491

Properties of a Page node . . . 492

Sub-nodes of a Page node . . . 493

Definition of a Transition node . . . 494

Syntax of a "Transition" node . . . 494

Properties of a Transition node . . . 494

Specificities of a Transition node . . . 495

- Why define transitions in the "Root" node? . . . 495

Definition of a Finish node . . . 496

Definition of a Start node . . . 496

Definition of Long and String nodes . . . 497

Definition of a Control node . . . 497

General syntax of a Control node . . . 498

Types of controls and associated properties . . . 498

- Common properties . . . 499
- The CheckBox control . . . 500
- The ComboBox control . . . 501
- The ListBox control . . . 502

The Label control 506
The OptionButtons control 507
The ProgressBar control 507
The CommandButton control 508
The DBListBox control 508
The DBQueryBox control 512
The DBEdit control 515
The DBTable control 516
The DBPath control 516
The LinkEdit control 517
The TextBox control 517
Example of creating a wizard 518
Step #1: Analyze your needs 518
Step #2: Define how the wizard is organized 519
Step°#3: Transcribe the structure of the wizard using the scripting language
. . . . 521
Using the graphical editor 525
Overview of the interface 526
Creating a new node 528
Editing the properties of a node 528
Compiling, executing and debugging a wizard 529
Frequently asked questions 530
Question 530
Answer 530
Question 530
Answer 530
Question 530
Answer 531
Question 531
Answer 531
Question 531
Answer 532
Question 532
Answer 532
Question 532
Answer 532
Question 532
Answer 533
Question 533
Answer 533
Question 533
Answer 533

Chapter 24. Tuning AssetCenter for use in a WAN environment 535

Options in the Tools/ Options menu item 535
Lists 537
Setting the parameters for a particular list 537
Setting list parameters at the database level 538
Setting list parameters at the AssetCenter client level 539
Stripping down screens 541
Connection cache 541
Access restrictions 541
Applying the configuration of one client to other clients 541

Chapter 25. Integrating external applications 543

InfraTools Remote Control 543
Available functions 543
Prerequisites 543
Taking control of a computer from AssetCenter 544
InfraTools Desktop Discovery 545
Available functions 545
Prerequisites 546
Scanning a computer from AssetCenter 548
Installing a full version of InfraTools Desktop Discovery 549
Knowlix 549
AutoCAD integration 550
Presentation of AutoCAD/AssetCenter integration 550
Installation 551
Using AutoCAD integration 553

List of Figures

2.1. Example of customizing the Assets table 38
2.2. Using intermediary tables 47
2.3. Link between the Assets table and the Fixed assets table 48
2.4. Links between the History table and a target table 49
4.1. Detail of an itemized list 65
6.1. User profile detail 76
6.2. User rights 81
6.3. User Profile: Access restrictions tab 83
7.1. Comparison of AQL and SQL 96
7.2. "CurrentUser" function used as variable in an action 103
7.3. Example in the table of departments and employees 105
7.4. Intended uses of the query editor 111
7.5. Expression builder 115
8.1. Form header 149
11.1. Detail of an executable type action 173
11.2. "LetterTemplate.Doc" 174
11.3. Detail of a messaging type action with a referenced object 178
12.1. Main screen of AssetCenter Server 196
12.2. Time table periods 202
12.3. Time table periods 202
13.1. Overview of internal messaging system 240
14.1. Simplified representation of a workflow scheme 258

14.2. Overview of workflow in AssetCenter 262
14.3. Main tables allowing you to define a workflow scheme 263
14.4. Main tables involved in a workflow instance being executed 263
14.5. Detail of an activity 265
14.6. Workflow scheme 268
14.7. Example of a workflow 269
14.8. Detail of a workflow 270
14.9. "Request for technical validation" action 271
14.10. "Request approved" action 272
14.11. "Request refused" action 272
14.12. "Milwaukee Calendar" calendar 273
14.13. "Request approval" workflow scheme 273
14.14. Detail of a workflow activity 274
14.15. Time limit of "Functional approval" activity 275
14.16. "Approval not carried out" alarm 276
14.17. "Request approved" activity 277
14.18. Activation parameters of start event 278
14.19. Workflow scheme 279
14.20. Current workflow instance for the purchase request 282
14.21. Detail of the task resulting from the functional approval activity . . 283
14.22. Example of a workflow role 287
14.23. Example of a workflow role 287
14.24. Parameters tab of a Database type action 301
14.25. Example of synchronous workflow scheme 308
14.26. Example of asynchronous workflow scheme 309
14.27. Workflow scheme with terminal event 310
15.1. Selecting the type of data to be imported 315
15.2. Defining how to parse text files (screen 1) 328
15.3. Defining how to parse text files (screen 2) 330
15.4. Describing fields in a source text file 332
15.5. Configuring how records are created in linked tables 339
15.6. Mapping source fields to target database fields 340
15.7. Using a linked field as key of the main table 346
15.8. Defining a field as a key in a linked table 347
15.9. Key updating values of characteristics 348
15.10. Modifying the content of a linked record for which there is no reconciliation key 350
17.1. The script builder 379
17.2. Parameters of the "Tutorial" feature 382
17.3. Parameters of the feature "Example1" 392
17.4. Editing the program 394
17.5. Editing the program 396
18.1. Timetables tab in calendar detail 400

18.2. The Exceptions tab in the detail of a calendar 402
18.3. Preview tab in calendar detail 405
19.1. First parameter 416
19.2. Second parameter 417
19.3. Database creation options 420
21.1. Detail of a calculated field 460
21.2. Example of a calculated field 464
23.1. Example 472
23.2. Structure of a wizard 475
23.3. Page of a wizard 476
23.4. Wizard organization 520
23.5. Execute and debug button 527

List of Tables

1.1. Database options 28
2.1. Prefixes for SQL names according to the data type of the field 38
2.2. Type of a field (**Type**) 40
2.3. Data entry format and data type 40
2.4. Value of the **User types**field 42
2.5. Various types of existing links 45
2.6. Nature of the information stored by links 45
2.7. HTML tags recognized in help on fields and links 55
2.8. Reserved characters in the HTML language 56
4.1. Values in system itemized lists 66
7.1. Example of sort order without indexes forced 109
7.2. Conventions 117
7.3. Logical operators 122
7.4. Comparison operators 123
7.5. Aggregate-type AQL functions 129
7.6. String-type AQL functions 130
7.7. Date-type AQL functions 131
7.8. Examples of Date-type AQL functions 133
7.9. Numeric-type AQL functions 134
7.10. Test type AQL functions 134
8.1. Editing forms and objects - icons 142
10.1. Advantages of using the ODBC driver 158

11.1. Example of a **Script** type action 179
12.1. AssetCenter Server program menus and icons 196
12.2. Days of monitoring 201
14.1. Limiting workflow instances in progress for a given object - different possible cases 285
14.2. Limiting workflow instances in progress for a given object - different possible cases 302
14.3. The different ways in which an event can be processed 303
14.4. The different ways in which an event can be processed 306
15.1. Constraints to be respected when importing data into a field according to the value of the "UserType" property of the target field 320
15.2. Constraints to be respected when importing data into a field according to the value of the "Type" property of the target field 322
15.3. Constraints to be respected when importing data into a field according to the value of other properties of the target field 322
15.4. How to populate the different fields shown above between <> . . . 357
17.1. List of types 380
17.2. Fields for populating the feature 382
17.3. Values used in a system itemized list 388
17.4. Analyzing and defining the algorithm - values used in a system itemized list 393
18.1. Values of the **Days** field 403
19.1. Format of Daylight time field 410
19.2. Values of the <DaylightInfo> argument 411
19.3. Different possible situations for the LocalTimeZone when connecting to a database 421
21.1. Calculated fields types 457
21.2. Correspondences 458
23.1. Conventions used 469
23.2. List of methods that can be used 481
23.3. "Table" type property 482
23.4. Example 483
23.5. Global variables features 483
23.6. Logical properties of the "Root" node 486
23.7. Physical properties of the "Root" node 489
23.8. Sub-nodes of "Root" node 490
23.9. Logical properties of a "Page" node 492
23.10. Physical properties of a "Page" node 492
23.11. Sub-nodes of "Page" node 493
23.12. Logical properties of a "Transition" node 494
23.13. Logical property of a Finish node 496
23.14. Logical property of a Long or String node 497
23.15. Logical properties common to all controls 499

23.16. Physical properties common to all controls 499
23.17. Property of "CHECKBOX" control 500
23.18. Physical properties of the "COMBOBOX" control 501
23.19. Physical properties of the "LISTBOX" control 502
23.20. Mandatory logical property of the "LISTBOX" control 505
23.21. Physical properties of the "LABEL" control 506
23.22. Physical properties of the "OPTIONBUTTONS" control 507
23.23. Physical properties of the "PROGRESSBAR" control 507
23.24. Physical properties of the "COMMANDBUTTON" control 508
23.25. Physical properties of the "DBLISTBOX" control 509
23.26. Physical properties of the "DBQUERYBOX" control 512
23.27. Mandatory physical properties of the "DBEDIT" control in "Normal" mode 515
23.28. Mandatory logical property of the "DBPATH" control 516
23.29. Mandatory logical property of the "LINKEDIT" control 517
23.30. Physical property of the "TEXTBOX" control 517
23.31. Definition of transitions 520
23.32. Page contents 520

CHAPTER 1

Database options

When you are connected as the administrator, the **Tools/ Database options** menu item enables you to configure certain options inherent to AssetCenter and/or the database.

Conventions

The edit screen displays the assigned values for each option, and the minimum and maximum values for the time span.

Values displayed in black cannot be edited.

The **Default value** indicates the minimum value defined by Peregrine Systems

You can modify the **Current value** of the option, which is displayed in blue. Specific controls are intended to help you edit the values of options (drop-down list for a "Yes/No" value, text control for entering a text string).

The **Value used** displays the default value, or the current value, if it has been modified by the user.

Important: Any modifications you make in the Current value column are only taken into account when you restart AssetCenter.

Table 1.1. Database options

Section	Name of the option	Description	Example of values	Editable
Agent	Deactivate history during import	If this option is set to **Yes**, the history lines associated with records are not imported.	No	Yes
Procurement	Have the application server (aamsrv) create the items received	If this option is set to **Yes**, the ordered items are received in the **amReceipt** table but are not created in the portfolio. The corresponding records are stored in the intermediary **amItemsReceived** table and processed by the **Create assets, consumptions, etc. corresponding to items received** module of AssetCenter Server.	No	Yes
Wizards	Configuration script	This script contains Basic functions that can be called on in **Wizard** type actions.	N/A	Yes
Tips of the day	Strings	List of the "Tip of the day" tips displayed when AssetCenter is launched;	TIP2, "Shift+F9 starts the wizard debugger."	Yes
Authorization	Signature	Used by AssetCenter to verify that AssetCenter Server regularly connects to the database.	Updated by AssetCenter Server.	No
Authorization	Signature file	License file that you inserted in the database.		No
Documents	Maximum size of documents that can be inserted into the database.	This size is expressed in bytes.	5 242 880	Yes
Feature	Deactivate consolidation of features while modifying a feature value	If this option is set to **Yes**, the value of a consolidated feature will not be recalculated when the value of at least one of the features making it up changes.	No	Yes
Access control	Slot update delay in minutes	Interval at which the client updates the login slot.	10	Yes
Access control	Slot timeout in seconds	Timeout period for the database connection. This timeout is only used if the automatic disconnection option is activated (value **Yes**).	1800	Yes

Section	Name of the option	Description	Example of values	Editable
Access control	Automatic disconnection	Activation of the automatic disconnection option.	Yes	Yes
Time zone	Server	Time zone of the server. The expected value is a numeric value specifying the difference with reference to GMT. An empty value means that no time zone is used. A null value (=0) defines the time zone as GMT.	4 (time zone GMT+4)	Yes
Time zone	Data	Time zone of data in the database. The expected value is a numeric value specifying the difference with reference to GMT. An empty value means that no time zone is used. A null value (=0) defines the time zone as GMT.	-2 (time zone GMT-2)	Yes
Event management	Expiration time for output events (in hours)	Output events are queues that can receive data and export it to external applications. The expiration time sets the value that determines when to purge these events.	336	Yes
Event management	Expiration time for input events (in hours)	Input events are queues that can receive data and import it from external applications. The expiration time sets the value that determines when to purge these events.	336	Yes

CHAPTER

2 Structure of the AssetCenter database

This section provides information required to understand the description of the database.

Note: Warning: You should never delete or modify a unique index, nor directly write in the database using external tools (via INSERT, DELETE, UPDATE statements or triggers). For the latter, we suggest you use the AssetCenter APIs.

Foreword

To import data, access the database using external tools, write queries, etc., you need to understand the structure of the database. You will need information such as: the names of the fields, their maximum length, format, and whether or not their values must be unique.

There are several ways to obtain a description of the AssetCenter database structure:

- **Database.txt** and **Tables.txt** files: They contain the complete structure of the database. These files are located in the **Infos** sub-directory of the AssetCenter installation directory.

> Note: These files describe the default database structure. It does not include any customization you may have performed.

- AssetCenter Database Administrator program: It enables you to freely create description files of the AssetCenter database (tables, fields, links and indexes).

  It makes use of:

  - A description file of the AssetCenter database (file with the **.dbb** extension) or a connection to the AssetCenter database.
  - A template (file with the **.tpl** extension) that describes which information should be generated. We provide you with standard templates, and you can create your own templates. Sophisticated templates enable you to create files in **rtf** or **html** format.

  > Note: Among the templates provided with the standard version of AssetCenter, one of them, Dbdict.tpl, enables you to export the customization data (including information on features, calculated fields, configuration scripts, etc.) from your database to a standard text file. Used with a source control tool, this description file can help you track customizations made to the database.

- The AssetCenter program.

Definition of a database

An AssetCenter database is a group of files containing all the information on the assets you are managing. For simple installations, these files are all located in the same directory, either on the local disk drive or on a network file server.

AssetCenter comes with a demonstration database.

With AssetCenter, you can create several databases and open them one-at-a-time per each user session. In addition, there can be several user sessions connected to a database at the same time. The program uses a transactional method to update the data in the database, which means that AssetCenter employs a sophisticated mechanism to manage updates and to ensure that they are carried out in a secure and optimized fashion.

Structure of the Database.txt and Tables.txt files

The database structure is described in the **database.txt** and **tables.txt** files located in the **Infos** sub-directory of the AssetCenter installation directory.

Note: These files were created using the AssetCenter Database Administrator software and the Dbase.tpl and Tables.tpl templates. These templates are located in the Infos sub-folder of the AssetCenter installation folder.

Here is the format of these files:

- Type: text
- Separator: tab
- Character set: ANSI

The easiest way to view these files is to open them in a spreadsheet.

We have included as much information as possible in these files. You may choose to hide or delete the information you do not require.

The **Database.txt** file has the following structure:

- One line per field, link or index.
- The tables are sorted by SQL name.
- For each table, the following information appears in succession:

  1. Fields
  2. Links
  3. Indexes

- Fields, links and indexes are sorted by SQL name.
- One data item per column. The first line in the table indicates the template parameters that were used to generate the information. The second line includes a clear description of the type of information generated.

The structure of the **Tables.txt** file is very simple:

- One line per table.
- Tables are sorted by SQL table name.
- One data item per column. The first line in the table indicates the template parameters that were used to generate the information. The second line includes a clear description of the type of information generated.

Note: For information on the parameters used to generate the Database.txt and Tables.txt files, refer to the manual entitled "Reference Guide: Administration and Advanced Use", chapter "Creating, customizing and describing the AssetCenter database", section "Description of an AssetCenter database".

Note: Note: The SQL names identify the tables, fields, links and indexes. These names do not change from one AssetCenter language version to the next; they are always the same.

Description of the tables

This section describes the tables in the AssetCenter database:

- Identification of AssetCenter tables
- String describing the AssetCenter tables

Identifying AssetCenter tables

Each table in the AssetCenter database is described by:

- An "SQL name". The SQL names of AssetCenter tables are in English and have the "am" prefix. These names do not change from one AssetCenter language version to the next; they are always the same.
- A "Label": This is the name of the table as it appears in AssetCenter.

  It is used when AssetCenter has little space available to display the table name (tree view of the database in filters and queries, error messages, menu names, etc.).
- A "Description": This description is used when AssetCenter has sufficient space available to display the entire description (status bar, for example).

Table description strings

Use AssetCenter Database Administrator to define the description strings used for AssetCenter tables.

In an AssetCenter table, the description string describes the records appearing in drop-down lists that enable you to select linked records.

It also defines the name that appears in the title bar of the window that displays the records.

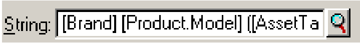
The description string can contain the following elements:

- Field SQL names, between brackets and square brackets.
- Text strings without text delimiters.

- Links at 1 or several levels.

Example :

Figure 2.1. Example of customizing the Assets table



String: [Brand] [Product.Model] ([AssetTa

In this table, the title of the detail window for an asset whose brand is "Asus", model is "AsusLX512" model and asset tag is "1" will be:

```
Detail of asset 'Asus AsusLX512 (1)'
```

When a drop-down list appears in list mode, it is sorted in ascending order, except if a filter is applied to the list by AssetCenter.

On the other hand, when a list appears in tree view, records are sorted in ascending order according to the **Full name**. The **Full name** is a field used in hierarchical tables. Its SQL name is "FullName"

Description of fields

This section describes the fields in the AssetCenter database:

- Identification of AssetCenter fields
- Field types and data entry formats
- dtLastModif fields
- FullName fields

Identifying AssetCenter fields

Each field in an AssetCenter table is described by:

- An "SQL name". SQL names are in English. SQL names do not change from one AssetCenter language version to the next; they are always the same. The SQL names are prefixed according to the data type of the field:

Table 2.1. Prefixes for SQL names according to the data type of the field

Prefix	Data type of the field
"b"	Boolean
"d"	Date
"dt"	Date+Time
"l"	Long integer
"m"	Monetary
"p"	Percentage
"se"	System enumeration (system itemized list)
"ts"	Duration (time span)
"mem"	Comment
None	Character string

- A "Label". AssetCenter uses labels when it has little space available to display the field name (detail screens, tree view description used when creating queries, etc.).
- A "Description", used to describe the field in the AssetCenter status bar.

Field types and data entry formats

This section lists:

- The type of AssetCenter fields.
- The data entry formats and data types for AssetCenter fields.

To view the type of a field, its data entry format and the type of its data, you can:

- Use the **Type** field in the **Detail** tab of the AssetCenter Database Administrator.
- Read the **Database.txt** file, located in the Infos folder of the AssetCenter installation folder. The **Field Type** column details the field types used.
- Display the context-sensitive help for that field.

Type of a field ("Type")

Table 2.2. Type of a field (Type)

Value given in Field Type column in Database.txt file	Value in Type field in AssetCenter Database Administrator	Signification
BYTE	Integer (8 bit)	Integer from -128 to +127.
SHORT	Integer (16 bit)	Integer from -32,768 to +32,767.
LONG	Integer (32 bit)	Integer from -2,147,483,647 to +2,147,483,646.
FLOAT	Floating point number	4-byte floating point number.
DOUBLE	Double-precision number	8-byte floating point number
STRING	Text	Limited-length text field. All characters are accepted.
BLOB	Binary field	Variable-length binary field (or BLOB Binary large object) can stores images and forms, for example, without size restriction.
MEMO	Long text field	Variable-length text field (comments). Certain DBMSs have limitations concerning these fields. Oracle for WorkGroups, for example, does not allow you to sort on this type of field.
DATE+TIME	Date and time	Date and time.
DATE	Date	**Date** format field only (no time).
TIME	Time	**Time** format field only (no date).

Provides the field's storage format.

Data entry format and data type

The data entry format and the data type are indicated by:

- The value of the **User type** in the **Detail** tab of the AssetCenter Database Administrator: By default, this value comes from the **Type** field.
- The value displayed for the field in the **Field data display and entry type** column in the **Database.txt** file.

You can nevertheless indicate the data entry type when it is verified:

Table 2.3. Data entry format and data type

Value given in Field data display and entry type column in database.txt file	Value in User type field in AssetCenter Database Administrator	Signification
Default	Default	The information is displayed and entered just like it is stored in the database, i.e. depending on the field's **Type**.
Numerical	Number	Number.
Yes/No	Boolean (Yes/No)	Boolean.
Money	Monetary	Monetary values.
Date	Date	This field can only contain **Date** type values.
Date+Time	Date+Time	A field with this data entry format contains **Date+Time** type values.
N/A	Time	This field may only contain **Time** type values
Time span	Duration	**Duration** type values. The authorized and default units are defined by the **UserTypeFormat** property.
System itemized list	System itemized list	You must select the value for this type of field from a "system" itemized list. This is called a "system" itemized list because the values are defined by the software and cannot be customized by the user.
Custom itemized list	Itemized list	You select the value for this type of field from an itemized list that may be customized by the user.
Percentage	Percentage	A field with this format contains percentages. This **Percentage** format displays values with two decimal places.
N/A	Feature value	Reserved. Do not use.
N/A	Basic script	Reserved. Do not use.
Table or field SQL name	Table or field name	SQL name of a table or a field.
N/A	Unknown	Any other type.

You can have additional information on field formats if you set the **User type** field in AssetCenter Database Administrator to one of the following values:

Table 2.4. Value of the User typesfield

Value of User type field in AssetCenter Database Administrator	Additional information

Time span	Display format.
	`%U1[l][d][%U2[l][d]]...[%Un[l][d]]`
	Syntax:
	`Ux`
	Where is one of the following units:
	• Y: year
	• M: month
	• D: day
	• H: hour
	• N: minute
	• S: second
	The optional "d" parameter specifies the default unit to be applied if no unit is entered. This parameter is assigned to one unit only.
	Example
	• "+HM,...H: displays hours, then minutes. By default, you enter hours.
	• "%YI%MId%DI": displays years, then months, then days.
	The optional "l" parameter specifies that the duration should be displayed in "long" format (e.g. 1998 is the long format of 98).

System itemized list	Values taken by the itemized list.
Itemized list	Name of the itemized list.
Object	Used internally by AssetCenter.

dtLastModif fields

The field with the SQL name "dtLastModif" exists in all tables in the AssetCenter database.

- SQL name: "dtLastModif".
- Short description: "Modified on".
- Long description: "Modification date".

This field is updated each time a record is modified in the AssetCenter database, whether this is via the user interface or an import operation. It indicates the date of modification or creation of the record. If you import a value in this field, that value is used instead of the actual import date.

"FullName" fields

The **FullName** field is a field in the hierarchical tables. Its SQL name is "FullName".

Structure of the "FullName" field

For each record in a hierarchical table, the "FullName" field stores the value of a field from that record. This value is preceded by a hierarchy, composed of the values of the fields of the parent records to which the record is linked, all the way down to the root.

Values are separated by the "/" character, without spaces.

This character also appears at the start and end of the hierarchy.

Examples:

- Location table: The full name of a location consists of the name of the location preceded by the name(s) of the parent location(s).

  ```
  Example : "/USA/Milwaukee/The Domes site/".
  ```

- Departments and employees table: The full name of an employee consists of the employee's last name, first name and ID, preceded by the name(s) of the parent record(s).

  ```
  Example : "/Commercial
  Services/Telemarketing/Colombo,Gerald,P223/".
  ```

- Assets table: The full name of an asset consists of its asset tag preceded by the asset tags of the parent asset(s).

  ```
  Example : "/P123/DD456/CM0125/".
  ```

Note: Warning: You cannot directly write in the "FullName" fields. They are entirely managed by AssetCenter.

Special case

If one of the values included in the value of a "FullName" field contains the "/" character, this character is replaced by "-".

Example: For departments and employees, if a department name is "Sales/Marketing", the "FullName" field for its components will have the following format: "/A.../Sales-Marketing/B.../".

Description of links

This section describes the links in the AssetCenter database:

- Identification of links
- Type of a link
- Cardinality of a link
- Intermediary tables
- Contextual links

Identifying links

A link is identified by:

- Its "SQL name". SQL names do not change from one AssetCenter language version to the next; they are always the same.
- Its "Label".
- Its "Description".

Example of the **Location** link from the Assets table:

- The SQL name of the source field in the source table (Assets table) is "lLocaId".
- The SQL name of the target table (Locations table) is "amLocation".
- The SQL name of the target field in the target table (Locations table) is "lLocaId".
- The "database.txt" file indicates that the link's cardinality is "1": A given asset can only have one location.

This link may be understood as: "source field = target field".

Warning: The label and description of the link are different from the label and description of the target table. This is because several links can exist between AssetCenter tables. For example, between the Assets table and the Departments and Employees table, the link whose label is User (SQL name: User) defines the asset's user, and the link whose label is Supervisor (SQL name: Supervisor) defines the asset's supervisor.

Type of a link

The following table lists the various types of existing links:

Table 2.5. Various types of existing links

Type	Description
Normal	If the source record is deleted, the link is deleted, and the references to the source record in the target records are emptied.
Own	If the source record is deleted, the link's target records are deleted.
Define	You cannot delete a source record if it is linked to target records.
Neutral	If the source record is deleted, the link is deleted. There is no information to update in the target records.
Copy	When the source record is duplicated, the links of this type are also duplicated.
Owncopy	The same meaning as "Own" and "Copy" links.

The following table lists the nature of the information stored by links:

Table 2.6. Nature of the information stored by links

Nature of the information	Description
Normal	The link stores something different from what is stored by the other "UserType" links.
Comment	The link stores a comment field.
Image	The link stores an image.
History	The link stores a history.
Feature value	The link stores a feature value.

Cardinality of a link

We have defined two types of links from a given table A in AssetCenter:

- "1-->1" links: A record in table A can only be linked to a single record in table B. For example, the Employees table is linked to the Locations table via a "1-->1" link: An employee can only be associated with one single location.
- "1-->N" links: A record in table A can be linked to several records in table B. For example, the Contracts table is linked to the Assets table via a "1-->N" link: A contract can concern several assets.

Important note: In theory, there are three types of logical links between tables in a database:

- 1 links: A record in table A can only be linked to a single record in table B, and vice-versa. A 1 link between two tables in AssetCenter is represented by two "1-->1" links.
- N links: A record in table A can be linked to several records in table B, but a record in table B is only linked to a single record in table A. An n link between two tables in AssetCenter is represented by a "1-->1" link and a "1-->N" link.
- N-n links: A record in table A can be linked to several records in table B, and vice-versa. An n-n link between two tables in AssetCenter is represented by two "1-->N" links.

Intermediary tables

Intermediary tables are only used in the case of n-n cardinality logical links.

They do not appear in the AssetCenter interface, which only shows the logical links between tables.

As opposed to normal tables, intermediary tables (sometimes called relation tables) do not have a primary key.

The following diagram explains how intermediary tables are used:

Figure 2.2. Using intermediary tables

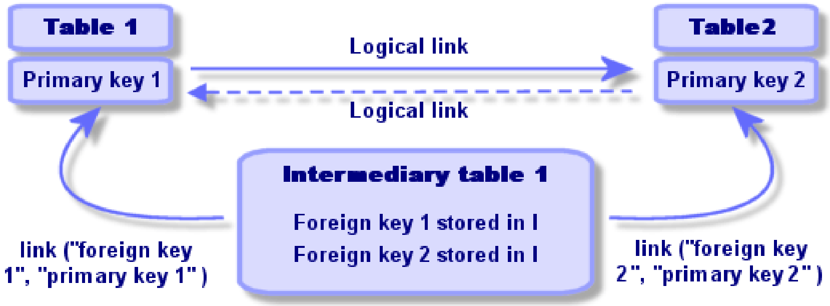
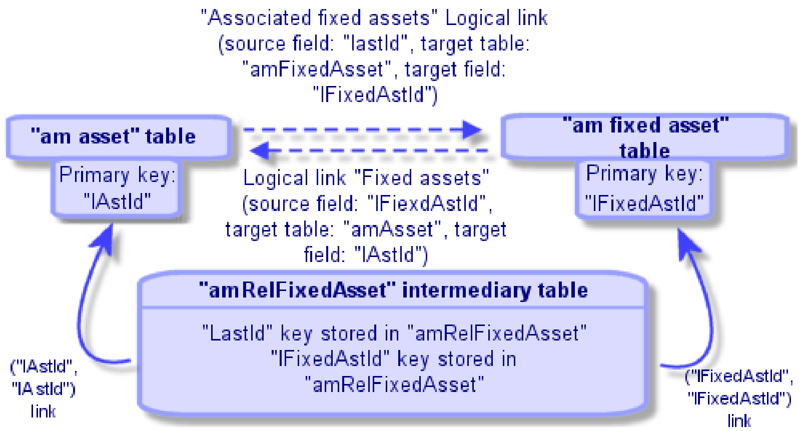


Table 1
Primary key 1
Logical link
Logical link
Table2
Primary key 2
Intermediary table 1
Foreign key 1 stored in I
Foreign key 2 stored in I
link ("foreign key 1", "primary key 1")
link ("foreign key 2", "primary key 2")

Example of the link between the Assets table and the Fixed assets table:

Figure 2.3. Link between the Assets table and the Fixed assets table


"Associated fixed assets" Logical link (source field: "lastId", target table: "amFixedAsset", target field: "lFixedAstId")
"am asset" table
Primary key: "lAstId"
"am fixed asset" table
Primary key: "lFixedAstId"
Logical link "Fixed assets" (source field: "lFiexdAstId", target table: "amAsset", target field: "lAstId")
"amRelFixedAsset" intermediary table
"LastId" key stored in "amRelFixedAsset"
"lFixedAstId" key stored in "amRelFixedAsset"
("lAstId", "lAstId") link
("lFixedAstId", "lFixedAstId") link

In this case:

- An asset can be the subject of several fixed asset items (logical link **Associated fixed assets**):
  - Each record in the Assets table can be linked to several records in the intermediary table.
  - Each record in the intermediary table is linked to one single record in the Fixed assets table.
- A fixed asset can concern several assets (logical link **Fixed assets**):
  - Each record in the Fixed assets table can be linked to several records in the intermediary table.
  - Each record in the intermediary tab is linked to one single record in the Assets table.

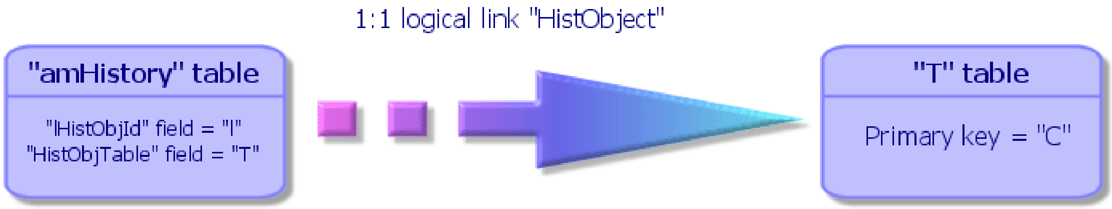
Note: Although intermediary tables do not appear in the AssetCenter user interface, it is sometimes necessary to use them when writing complex queries.

Contextual links

In some cases, the target table is not pre-defined but is specified in the source table. This is called a "contextual link". This kind of link, which has a cardinality of 1, has no reverse link.

Example: The case of a link between the History table and a target table:

Figure 2.4. Links between the History table and a target table



Description of indexes

This section describes the indexes in the AssetCenter database:

- Identification of indexes
- Uniqueness of field values of an index in a table

Identifying indexes

An index is identified by:

- Its "SQL name", ending with "Id". SQL names do not change from one AssetCenter language version to the next; they are always the same.
- Its "Label".
- Its "Description".

Uniqueness of indexed field values

The values of indexed fields may or may not be duplicated according to the nature of the index.

In AssetCenter Database Administrator, the nature of the index is shown by an icon displayed to its left:

- No icon: No constraints.
- 🔑: The indexed list of fields must not occur more than once in the table.
- ○🔑: The indexed list of fields must not occur more than once in the table, except for the "NULL" value, which may be repeated any number of times.

For example:

In the **Software** table (SQL name: amSoftware), the "Soft_PublisherName" index uses the following fields:

- **Name** (SQL name: Name)
- **Publisher** (SQL name: Publisher)
- **Version level** (SQL name: VersionLevel)

This index is "unique or null". This means that a record such as:

```
Microsoft, Word, 97
```

Cannot be found more than once.

On the other hand, it is possible to have duplicates of a record for which these three fields are "null" at the same time.

Customizing the database

AssetCenter enables you to customize the database in order to tailor it to the needs of your enterprise.

This level of customization is restricted to an administrator.

Modifications are visible to AssetCenter users. All users view the database in the same way, as defined by the administrator. When the administrator modifies the name of a field or a link, the new name appears whenever it is used in AssetCenter, in particular in the list screens, detail screens, and when building queries.

This section provides a detailed explanation of how to customize the various objects in the AssetCenter database:

- Customizing tables
- Customizing fields and links
- Customizing the indexes
- Default values for fields and links
- Counters in field default values
- Errors in the description of default values
- HTML tags recognized in help on fields and links

Customizing tables

You can customize a table's **Description**, **Label** and **Description string** using AssetCenter Database Administrator.

Customizing fields and links

There are several ways to customize the database fields and links:

- From the **Configure object** shortcut menu:

  To display the **Configure object** shortcut menu, move to the field to configure and right-click.

Note: If you use the shortcut menu, changes are recorded: either when you close the database and confirm the validation message, or when you select the Administration/ Save database configuration menu item.

- By using AssetCenter Database Administrator.

The **Detail** and **Scripts** tabs in AssetCenter Database Administrator, as well as the **General** tab allow you to define the:

- Labels for fields and links.
- Descriptions of fields and links.
- Default values for fields and links.
- Mandatory fields and links.
- Fields and links for which you want to log modification history.
- Read-only fields and links.

Note: You can also set the maximum size of a field of type "Text" via AssetCenter Database Administrator when creating the database.

- You can also modify the context-sensitive help for a field or a link from the **Help text** tab:
  - "Description": explains the contents of the field or link.
  - "Example": provides examples of how to populate the field or link.
  - "Important": list of "sensitive" points: data entry precautions, related automatic mechanisms, etc.

  The help system for fields is a sub-set of the HTML language.

Customizing the indexes

You can customize the indexes in the database using AssetCenter Database Administrator.

For each index, you can define the:

- Label.
- Description.

Default values for fields and links

Default values for fields and links are built from a series of:

- Constant values surrounded by "quotation marks".
- Script functions returning values.
- References to other database fields.

Note: AssetCenter automatically applies default values when creating a new record. Users creating or modifying the record can modify these values.

Example

```
RetVal="BL"+AmCounter("Delivery", 2)+AmDate()
```

- AmDate() provides the date on which the record was created.
- AmCounter("Delivery, 2) provides the current value of the "Delivery" counter incremented by 1, and expressed in two digits.

In the case of calculated fields

Calculated string and Basic script type calculated fields may only be used in the calculation of the default value of a standard fields. [#ValDefCasPartic]

In the case of links going to the Comments table (SQL name: amComment)

It is not possible to assign a default value to the links going to this table.

Example
Comment (SQL name: Comment)

Be careful when defining default values for fields and links

The calculation of default values for fields does not take access restrictions into account. You must therefore only reference those fields and links that may be viewed by all users.

Counters in field default values

When you define a default value for a field, you can reference a counter.

When creating a new record containing this field, AssetCenter automatically inserts the number, incremented by one unit, for each new record.

Counters are managed by the administrator using the Administration/ Counters menu item.

You must have created a counter using this menu before you can insert the counter in a formula for a field default value. Counter names cannot contain the following characters: space, "$", "(" or .")"

You can create as many counters as you want.

AssetCenter users never see the counter names.

The counter is incremented as soon as you click the New button. If you cancel the record creation without clicking the Create button, the counter is not decremented.

You are free to readjust the values of the counters.

The value displayed in the counter detail is the last number used for this counter.

Note: If you enter a formula such as AmCounter(<Counter name>, [n]) in the default value of a field instead of AmCounter(<Counter name>), the value of the counter will be displayed using n digits.

Errors in the description of default values

Here follow some of the most common mistakes made when describing default values:

You have defined a tag that must be unique in the default value

Some fields do not accept several variables. This is the case for date fields, for example. In this case, you can only enter one variable in the default value.

The type is not compatible with the tag

In some cases the variable may be incompatible with the field type. This is the case if you try to define AmLoginName() on a field of type date, for example.

Field XXX is not found in table XXX

When you refer to a link, you must use the "Link.Link.Field" format where each link is referenced by name. This lets AssetCenter directly follow the links.

For example, to set a default supervisor for an asset, you may replace the cost center supervisor with: "CostCenter.Supervisor".

The default value will be used as soon as the AssetCenter user populates the asset's cost center field. Then the supervisor can be located in the database.

HTML tags recognized in help on fields and links

The in-depth help system for AssetCenter fields uses a sub-set of the HTML language. Only a few "tags" are recognized, used mainly for formatting the ToolTip. This section does not attempt to provide an exhaustive description of the HTML language. It simply gives a brief description of each of the HTML tags managed by AssetCenter.

Note: In the following table, the HTML tags are listed as they must be entered by the user. They do not follow the notation conventions applicable to the rest of the online help.

Table 2.7. HTML tags recognized in help on fields and links

HTML Tag	Description
<FONT FACE="font name">	Defines the font used to display the text following this tag. This font remains in use until the next change of font.
<FONT COLOR=#RRGGBB>	Defines the font color used to display the text following this tag. This color remains in use until the next change of color.
<FONT SIZE=+n>	Increases the font size by "n" levels. This font size remains in use until the next change of font size.
<FONT SIZE=-n>	Decreases the font size by "n" levels. This font size remains in use until the next change of font size.
<B> </B>	These two tags surround text to be displayed in bold.
<I> </I>	These two tags surround text to be displayed in italics.
<LI>	This tag is the start of a bulleted list. It causes a carriage return and inserts a bullet at the start of the text.
<HR>	Draws a horizontal line used as a text separator.

For further information on the HTML language, refer to one of the numerous publications on the subject.

Reserved characters

The following table lists the reserved characters in the HTML language. If you enter these characters, they will not be displayed as such on the screen. You should use the corresponding tag to display the desired character.

Table 2.8. Reserved characters in the HTML language

Reserved character	Use the following tag to display this character
"<"	"&lt;"
"&"	"&amp;"
Non breaking space	"&nbsp;"

3 Creating an AssetCenter database

CHAPTER

Here are the different steps in creating a database:

1 Declaring the connection to an empty shell, which itself is created using a database manager.

Note: You must create an empty shell in order to create a database. This task varies according to the DBMS used and is usually the job of an administrator. How to perform this task is beyond the scope of this document. For further information, consult the documentation of your DBMS.

2 Entering the authorization string. The authorization string determines your access rights, which correspond to your AssetCenter license agreement.

3 Entering or modifying the administrator password. The administrator login and associated password are initially defined by the DBMS used to create the empty shell.

4 "Physically" creating the AssetCenter database. During this step, AssetCenter creates the full structure of the database (tables, fields, links, etc.) inside the empty shell.

5 Starting AssetCenter Server and connecting to the database. This step validates the authorization string and enables users to connect to the database.

Declaring a connection to an empty shell

Here are the steps to follow to declare a connection to an empty shell:

1 Start AssetCenter.

2 Select **File/ Manage connections**.

3 Click New.

4 In the **Connection** tab, populate the **Name** field, the DBMS, and the information that is specific to this DBMS.

5 Click Create to create the connection.

Entering the authorization string

To activate the database access rights corresponding to your AssetCenter license, you must provide AssetCenter with an authorization string. This string is stored definitively in the database; it does not need to be declared at the level of client machines. Each authorization string is associated with a unique security key. The authorization string and key allow AssetCenter to validate your user rights.

This key can be obtained from Peregrine Systems after providing certain pieces of information. The pieces of information needed by Peregrine Systems together with the field used to enter the authorization string are located in the window that will be displayed just after launching the creation of the database.

Before calling Peregrine Systems:

1 Define the machine to be used to run the instance of AssetCenter Server that will regularly verify the authorization string. This is the machine on which the **Signal presence to database server** module is activated (**Options/ Configure** menu item, **Modules** tab). If possible, this instance of AssetCenter Server should not be moved to another computer, since this will require changing the authorization string.
2 Determine the Mac address of the network card of this computer.

Note: To find the Mac address of the network card of a given machine, you can execute AssetCenter Server on this computer: Select Help/ About and click . You do not need to connect to a database to do this.

Warning: If AssetCenter Server runs on a computer under Windows 95, then you must install the netbevi protocol. This protocol allows AssetCenter Server to correctly identify the Mac address of the network card.

Note: We recommend that you run AssetCenter Server on a computer equipped with Windows NT.

The authorization string determines:

- The number of authorized users.
- The maximum number of assets and main assets that you can create.
- The DBMS that may be used.

- The enabled functions.

Here are the steps you must follow:

1. Once you have declared the connection via the **File/ Manage connections** menu, click Test to create the database.
2. When the authorization string dialog box is displayed, call Peregrine Systems.
3. Tell the person at Peregrine Systems the value of the **MAC address for AssetCenter Server** field (this value is displayed automatically), and the value of the **Registered name (company)** field (you must enter this value).
4. Enter the authorization string provided by the person at Peregrine Systems in the **Authorization string** field.

Note: If you change the network card of the computer on which AssetCenter Server is executed, you will need to call Peregrine Systems for a replacement authorization string.

Once the authorization string is validated, AssetCenter displays the database creation dialog box. If authorized to do so by your license agreement, you can declare time-zone information here. To do this, select the **Use time zones** option and populate the **Server time zone** and **Data time zone**.

To continue creating the database, click Create.

Entering the administrator password and "physically" creating the database

At the start of the database creation process, you must enter the administrator password. An administrator login and password are also defined when you create an empty shell using the database manager.

AssetCenter allows you to change the administrator password before creating the database structure. The time taken to perform this operation varies according to the DBMS and the speed of the computer used.

Once the creation is complete, start AssetCenter Server and connect to the database to validate the database in order for users to be able to connect.

Modifying the authorization string

If you change the network adapter or computer on which AssetCenter Server runs, or if your license expires, you must change the authorization string in your database. You do not have to be connected to the database to carry out this modification.

Modifying the authorization string using AssetCenter Database Administrator

1. Start AssetCenter Database Administrator.
2. Connect to the database whose authorization string you want to modify.
3. Select the **Action** menu and then **Edit license file**. Enter the new license file in the dialog box that appears.

4 Managing itemized lists

CHAPTER

An itemized list is a list of values proposed by AssetCenter for populating certain fields (standard fields in a detail screen, or the value of a feature), e.g. quality, position, country, brand.

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

The values appear in a "drop-down list". Simply choose the correct value from the list to assign a value to the field.

AssetCenter manages two types of itemized lists:

- Free itemized lists
- System itemized lists

Free itemized lists

The AssetCenter administrator can access free itemized lists using the Administration/ Itemized lists menu item.

There are two types of free itemized lists:

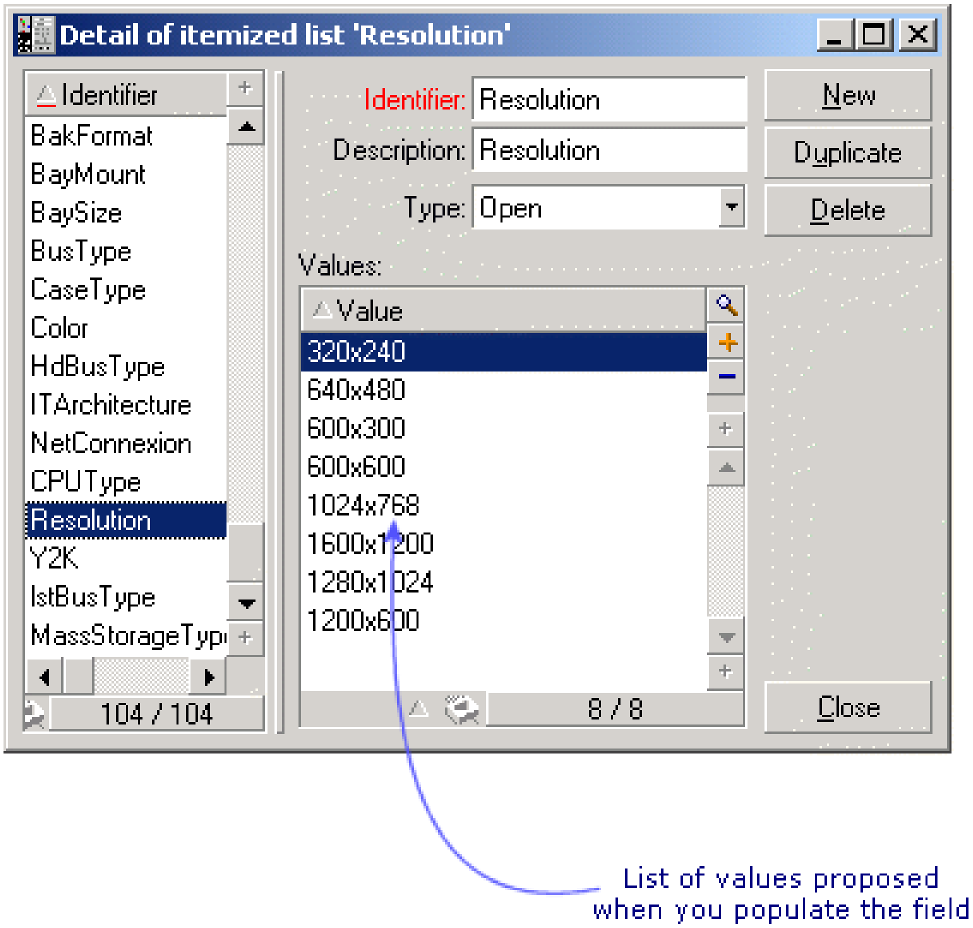
- Itemized lists that you create yourself. You can link them to features, but you cannot attach them to fields: Only the software can assign an itemized list to a field.
- Lists that are linked to database fields. These lists may also be attached to features. If you delete an itemized list of this type, or modify its name, AssetCenter will recreate the itemized list (with no associated values) using its original name, as soon as the itemized list is required to create a record containing a field which should be linked to that list.

Values of 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.

The administrator can delete, modify or add values using the ⊟, 🔍 and ⊞ buttons located to the right of the list.

Figure 4.1. Detail of an itemized list



Detail of itemized list 'Resolution'

Identifier: Resolution

Description: Resolution

Type: Open

Values:

Value
320x240
640x480
600x300
600x600
1024x768
1600x1200
1280x1024
1200x600

Open itemized lists

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

AssetCenter 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 itemized lists

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

AssetCenter users enter only values included in this list.

Note: After clicking or in the itemized lists management window, any modifications that the administrator defines for itemized lists are stored in the database. Modifications made to itemized list are only taken into account at the client level after the next connection to the database following the modification.

System itemized lists

The list of values in a system itemized list is defined by AssetCenter. It cannot be customized by the administrator or a user.

These itemized lists cannot be edited from the **Administration/ Itemized lists** menu item.

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.

Table 4.1. Values in system itemized lists

Value stored in the database	Value displayed
0	**In used**
1	**In stock**
2	**Retired asset**
3	**Awaiting delivery**

Example of the **Assignment** field (SQL name: seAssignment) in the asset detail:

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

- Using the context-sensitive help for the field populated by the system itemized list.
- From AssetCenter Database Administrator.
- Using the **database.txt** file describing the database structure.

5 How the history function works

CHAPTER

Any modifications made to field values and table links in the database can be recorded. Every time you create, modify or delete a value in a field for which the history is kept, AssetCenter 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. From the shortcut menu, select **Configure object**.
2. Go to the **General** tab in the configuration screen.
3. Set the **History** field to **Yes**.
4. Click OK to confirm.

Note: Any modifications concerning history are saved in the database as soon as you click in the database customization window. You can also define whether history is kept for a field or a link using AssetCenter Database Administrator.

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

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** (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 **Configure list**.

Note: If you are importing a database from an older version of AssetCenter, 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).

Depending on the case, AssetCenter functions differently:

Record creation

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

AssetCenter records:

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

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

AssetCenter records:

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

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

AssetCenter records:

- **Modified on**: Date on which the addition was made.
- **Author**: Author of the addition.
- **Field**: 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

AssetCenter records:

- **Modified on:** Date on which the deletion was made.
- **Author**: Author of the deletion.
- **Field**: 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

AssetCenter 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

In AssetCenter, 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

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) box is checked.

AssetCenter 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:** 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**.

AssetCenter records:

- **Modified on:** date when the feature was deleted.
- **Author:** the person who deleted the feature.
- **Field:** 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**.

AssetCenter records:

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

Warning: If you delete a record, all history lines are also deleted, either straightaway or via AssetCenter Server.

Creating, deleting or modifying history lines

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

Creating 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.

AssetCenter 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**, AssetCenter automatically creates history lines for this feature. History lines may be viewed in the **History** tab of the table associated with this feature.

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

6 Managing user access to AssetCenter databases

CHAPTER

This chapter explains how to manage user access to AssetCenter databases.

Note: Access rules can only be managed by database administrators.

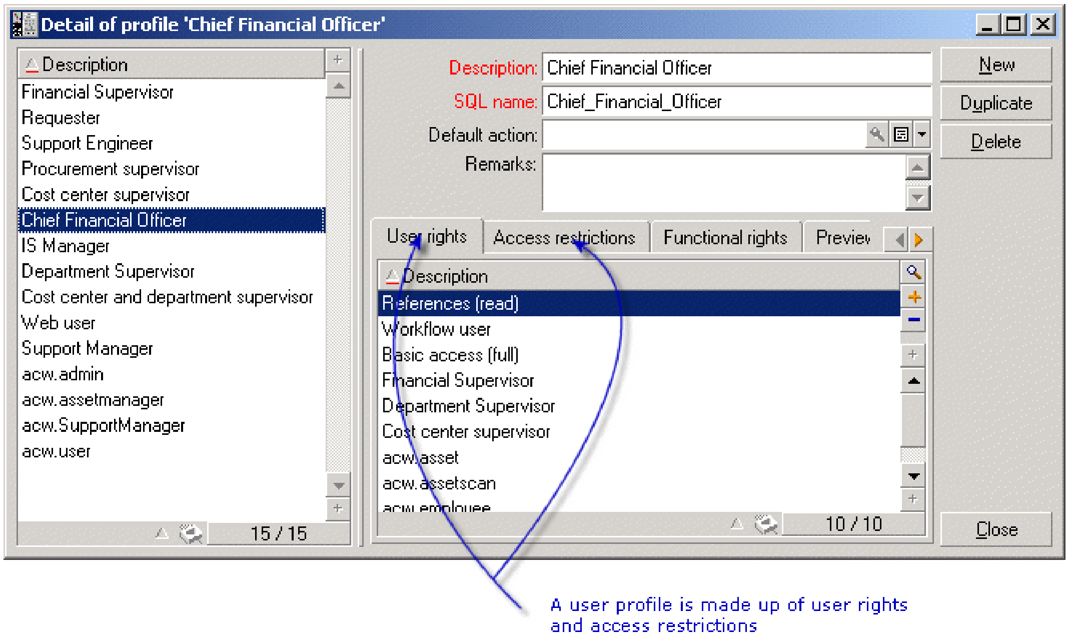
Use the Administration/ Rights management/ User profiles menu item to display the list of user profiles.

Use the Administration/ Rights management/ User rights menu item to display the list of user rights.

Use the Administration/ Rights management/ Access restrictions menu item to display the list of access restrictions.

Use the Administration/ Rights management/ Functional rights menu item to display the list of functional rights.

Figure 6.1. User profile detail



Detail of profile 'Chief Financial Officer'

Description
Financial Supervisor
Requester
Support Engineer
Procurement supervisor
Cost center supervisor
Chief Financial Officer
IS Manager
Department Supervisor
Cost center and department supervisor
Web user
Support Manager
acw.admin
acw.assetmanager
acw.SupportManager
acw.user

15 / 15

Description: Chief Financial Officer
SQL name: Chief_Financial_Officer
Default action:
Remarks:

User rights | Access restrictions | Functional rights | Previe

Description
References (read)
Workflow user
Basic access (full)
Financial Supervisor
Department Supervisor
Cost center supervisor
acw.asset
acw.assetscan
acw.employee

10 / 10

A user profile is made up of user rights and access restrictions

Importance and overview of database access rights

AssetCenter is a program that can be used simultaneously by several users: It uses a "shared" database.

Managing the database access rights in AssetCenter consists of two tasks:

1. Defining the data that each user can access, and the access conditions.
2. Managing user connections to the database according to the type of AssetCenter license acquired.

Defining access conditions for each user

All users are not necessarily able to view or modify the same data in the database. This depends on their position and the organization of the enterprise. One employee may be authorized to create assets, another to access stocks, another to access work order slips, etc.

In order to allow an employee to access AssetCenter:

- The employee must be included in the table of departments and employees.
- The Login field must be populated (Employees and Services table, Profile tab).
- The administrator must assign a "user profile" to the employee.

A user profile consists of user rights, functional rights and access restrictions.

A user right, a functional right or an access restriction can be used in several user profiles. A user right or functional right can be used in several user profiles. One user profile may be attached to several users. A given user can only have one profile, however.

Managing user connections

The AssetCenter license you acquired limits the number of database connections. The license may be by the "number of simultaneous users" or by the "number of declared users".

AssetCenter assigns a connection slot to each connected user.

AssetCenter automatically manages the database connection slots. Nevertheless, an AssetCenter administrator can also manage them.

How to ensure data security and confidentiality

To ensure data security and to make sure that your information is not wrongfully viewed, modified or destroyed, you can control three levels of access security:

- Define user access to the network.
- Define AssetCenter user profiles.
- Make regular backups of the database.

Definitions

This section defines the concepts related to access management.

- Definition of a user profile.
- Definition of a user right.
- Definition of a functional right
- Definition of an access restriction.

Definition of a user profile

A user profile is a set of user rights for given tables and fields and access restrictions that apply to records within these tables.

Profiles are attributed to users of AssetCenter.

For example, you can define:

- An "Accounting" profile, with access restricted to cost centers, budgets and expense lines only.

Definition of a user right

A user right is one of the component parts of a user profile. It concerns tables and fields in AssetCenter and not only certain records. As an administrator, you can assign read/write rights (as with an operating system) to different database users that apply to the different tables in AssetCenter.

Definition of a functional right

A functional right is one of the component parts of a user profile. It concerns categories of functionalities (such as Procurement, Cable and Circuit, etc.) and is based on functional domains. As an administrator, you can assign rights to users according to their jobs and to what directly relates to them and their fields.

Definition of an access restriction

An access restriction is part of a user profile. It corresponds to a record filter on a table. For example, you can make sure technicians only have access to the assets in their own department. Access restrictions concern both reading and writing (append or modification) records.

Defining access conditions

This section describes how to define access conditions:

- Defining user profiles.
- Defining user rights.
- Defining functional rights.
- Defining access restrictions.

Once defined, these access conditions are associated with AssetCenter users.

Defining user profiles

Use the Administration/ Rights management/ User profiles menu item to define user profiles.

Each "User profile" brings together:

- User rights for reading, writing, creating or deleting information in database tables or fields.
- Access restrictions, defining read and write conditions for records in a given table. For example, a technician can only view equipment located at the site where they work, or where a group of users works.

A user profile can be considered as concerning a given function in a company and its specific prerogatives.

Defining user rights

User rights can be managed via:

- The **Administration/ Rights management/ User rights** menu item.

- Or by clicking the ⊞ button to the right of the "User rights" list in a user profile detail (**Administration/ Rights management/ User profiles** menu item).

User rights describe the access to database tables and fields.

We suggest you create a user right per table that describes the rights concerning the table's direct and linked fields. You can set several access levels for a single database table:

For example:

- Basic accounting
- Advanced accounting
- Status access for employees
- Maintenance
- Etc.

You can then combine these user rights to create user profiles.

- Accountant
- Maintenance technician
- Intern
- Etc.

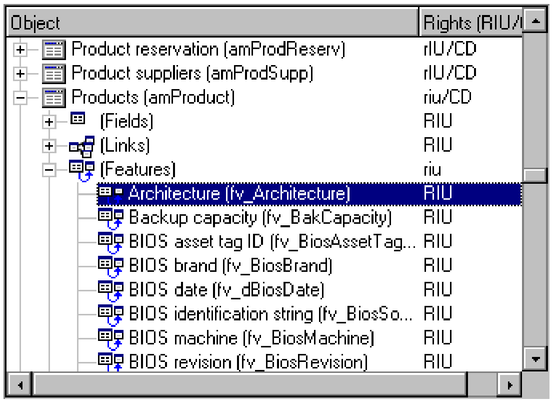
Editing user rights

The database structure is presented as a tree of tables. For each table, the tree contains a list of fields and features specific to the table, and a list of fields and tables from linked tables. Assign rights specific to tables, rights specific to fields, and rights specific to features.

The user right detail screen is organized as follows:

- Rights are displayed in a column.

Figure 6.2. User rights



Object	Rights (RIU/
Product reservation (amProdReserv)	rIU/CD
Product suppliers (amProdSupp)	rIU/CD
Products (amProduct)	riu/CD
(Fields)	RIU
(Links)	RIU
(Features)	riu
Architecture (fv_Architecture)	RIU
Backup capacity (fv_BakCapacity)	RIU
BIOS asset tag ID (fv_BiosAssetTag...	RIU
BIOS brand (fv_BiosBrand)	RIU
BIOS date (fv_dBiosDate)	RIU
BIOS identification string (fv_BiosSo...	RIU
BIOS machine (fv_BiosMachine)	RIU
BIOS revision (fv_BiosRevision)	RIU

- The **Display only tables with rights** filter allows you to view only tables for which user rights have been defined.
- When you select a node (Tables, Fields, Links, Features, etc.), AssetCenter automatically selects all the branches in the tree; this authorizes you to edit user rights for the entire node. For a parent node:
  - A lowercase "r" indicates that certain items in this node have read access rights.
  - An uppercase "R" indicates that all the items in this node have read access rights.
  - A lowercase "i" indicates that certain items in this node have create access rights.
  - An uppercase "I" indicates that all the items in this node have create access rights.
  - A lowercase "u" indicates that certain items in this node have update rights.
  - An uppercase "U" indicates that all the items in this node have update rights.
- You can perform multiple selections in this list. Therefore you can edit user rights for several items at once: Use the SHIFT and CTRL keys in to select multiple items in the tree.

Note: In order for a user to be able to see a "conditional" tab, they must have at least the right to view the fields that determines whether this tab is shown or not. For example, if a user does not have the right to view the Nature of payments field (SQL name: sePayType) in the General tab of a contract detail, they will not be able to see the Rents and Loans tabs in the contract detail, since these tabs are displayed or hidden depending on the value of the Nature of payments field.

Defining functional rights.

Functional rights can be managed via:

- The **Administration/ Rights management/ Functional rights** menu item.
- Or by clicking the ⊞ button to the right of the "functional rights" list in a user profile detail (**Administration/ Rights management/ User profiles** menu item).

A functional right defines the functional domain in which a user can operate. The Functional rights screen proposes a complete hierarchical list of the functional domains that you have defined. Double-click an item in this list to add or subtract it from the functional right.

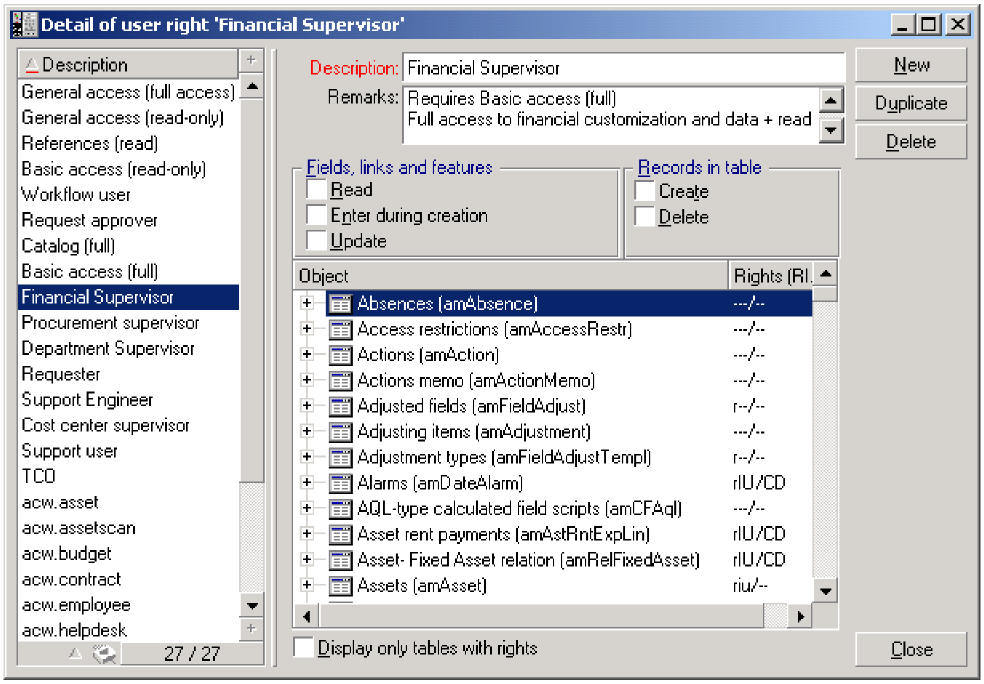
Defining access restrictions

Access restrictions can be managed via:

- The **Administration/ Rights management/ Access restrictions** menu item.
- Or by clicking the ⊞ button to the right of the "access restrictions" list in a user profile detail (**Administration/ Rights management/ User profiles** menu item).

An access restriction describes that records the user can see in the different tables of the database.

Figure 6.3. User Profile: Access restrictions tab



Detail of user right 'Financial Supervisor'

Description

- General access (full access)
- General access (read-only)
- References (read)
- Basic access (read-only)
- Workflow user
- Request approver
- Catalog (full)
- Basic access (full)
- Financial Supervisor
- Procurement supervisor
- Department Supervisor
- Requester
- Support Engineer
- Cost center supervisor
- Support user
- TCO
- acw.asset
- acw.assetscan
- acw.budget
- acw.contract
- acw.employee
- acw.helpdesk

27 / 27

Description: Financial Supervisor

Remarks: Requires Basic access (full)
Full access to financial customization and data + read

Fields, links and features

□ Read
□ Enter during creation
□ Update

Records in table

□ Create
□ Delete

Object	Rights (RI.
Absences (amAbsence)	---/--
Access restrictions (amAccessRestr)	---/--
Actions (amAction)	---/--
Actions memo (amActionMemo)	---/--
Adjusted fields (amFieldAdjust)	r--/--
Adjusting items (amAdjustment)	---/--
Adjustment types (amFieldAdjustTempl)	r--/--
Alarms (amDateAlarm)	rIU/CD
AQL-type calculated field scripts (amCFAql)	---/--
Asset rent payments (amAstRntExpLin)	rIU/CD
Asset- Fixed Asset relation (amRelFixedAsset)	rIU/CD
Assets (amAsset)	riu/--

□ Display only tables with rights

You can limit status access and or modification rights by using criteria similar to queries. These criteria may be used for example to cover:

- Categories, brands, or asset models.
- Departments or locations.
- Insurance contracts.

You can then combine access restrictions with user right to create user profiles:

- Maintenance engineer in St. Louis.
- Purchasing supervisor.
- And so on.

Editing access restrictions

Select the table concerned by the access restrictions and then define the access restrictions, for reading or for writing. These conditions are defined using the AssetCenter query editor.

Read condition

Allows the user to view records that satisfy the criteria defined in the query editor, excluding all other records.

Write condition

Allows the user to modify the field in an already existing record.

Warning: You must create a corresponding user right. This is not done systematically by AssetCenter.

Be careful when defining default values for fields and links

The calculation of default values for fields does not take access restrictions into account. You must therefore only reference those fields and links that may be viewed by all users.

Defining AssetCenter users

To define a new user:

1. Create the user in the table of departments and employees.
2. Move to the **Profile** tab in the employee detail.
3. Enter the employee's **Login** name and associated password. The **Login** is the name with which the user opens the database. This can be any character string.

Note: If you do not specify a password, the password is automatically the same as the Login. Users can then modify their passwords, after opening the database using the Login name.

4. Populate the **Login type** (SQL name: seLoginClass) field. Refer to the following paragraph to see the different options.

The remainder of the procedure depends on whether or not you want to assign administrative rights to this employee.

Defining a database administrator

Simply check the **Administration rights** box (SQL name: bAdminRight) in the **Profile** tab of the employee's detail. Then the employee will have all rights for all AssetCenter database tables and for the configuration of the database.

Note: There is a default administrator in the table of departments and employees: this is the "Admin" Login record. For the first AssetCenter installation, this is the only name available for accessing the AssetCenter database for all administrative operations.

Defining a non-administrative user

In this case, select a user profile in the **Profile** (SQL name: Profile) field in the **Profile** tab in the employee detail.

Managing user connections

This section explains how to manage user connections to the database.

- Three types of database access
- Principle of connection slots
- Destruction of connection slots

Database access types

AssetCenter defines several types of access to the database.

Define login types via the Profile tab in the employee detail.

Login types govern connections made to the database via the user interfaces of the Windows versions of AssetCenter and AssetCenter Web, or via the APIs.

Note: The login types do not take into account connections to the database made via AssetCenter Export, AssetCenter Server, AssetCenter Database Administrator.

Floating access

In this case, the license defines a maximum number of concurrent connections to the database; you cannot exceed this number.

You can define as many **Login** names as you want, but users may not be able to connect if the maximum number of concurrency users has been reached.

It is also possible for several connections to the database to use the same **Login**, but each connection decrements the remaining number of free connections.

Named access

In this case, the license defines the maximum number of **Login** names to the database that you can define.

Contrary to the previously defined mode, all declared users can connect to the database whenever they want.

However, you cannot enter more **Login** names than authorized by the license: AssetCenter displays an error message.

Furthermore, it is not possible for several connections to the database to use the same **Login** name.

Note: The access type of the "Admin" Login is Named. This user is not deducted from the number of authorized users.

Casual access

This login type is intended for users who rarely access the database. **Casual** users have standard logins and passwords, but their rights are limited.

This login type is restricted to users accessing the database via AssetCenter Web.

For example, a **casual** user can:

- View the assets they use.
- Create purchase requests.
- Follow the evolution of purchase request that concern them.

The rights of a casual user are limited:

- By their assigned user profiles.
- By a set of access restrictions defined at the level of AssetCenter.

These two elements are linked by an AND clause.

The number of casual users is not counted off.

Several concurrent connections to the database can use the same **Login**.

How connection slots work

When a user connects to the database via AssetCenter or AssetCenter Web, AssetCenter assigns a connection slot.

For as long as the user is connected to the database, the connection slot is regularly updated by AssetCenter. The frequency of updating is defined by the **Slot update delay in minutes** option in the table displayed by the **Tools/ Database options** menu item. By default, it is set to 5 minutes.

As soon as a user disconnects from the database, the corresponding connection slot is destroyed.

Detail of a connection slot

The AssetCenter administrator can view connection slots via the Administration/ Connection slots menu item.

A connection slot is defined by:

- The name of the application that is using it (in general AssetCenter, etc).
- The user's **Login**.
- The user's **Login type** (SQL name: seLoginClass).

Destroying connection slots

Connection slots can be destroyed:

- Manually via the Delete button in the connection slot management window.
- Automatically in the case of accidental termination of the application.
- Automatically if the connection is left idle.

Destroying connection slots manually

The administrator using the "Admin" login can force the disconnection of a user. To do this:

1. Display the connection slot screen via the **Administration/ Connection slots** menu item.
2. Select the appropriate connection slot.
3. Click Delete.

Note: This is only possible under the "Admin" login and not under other administration logins.

Detecting accidental termination of the application

In some cases, the application that uses a connection slot may be terminated accidentally (brutal shut-down of the operating system, network problem, etc.). Therefore users are not logged off from the database in the normal fashion (e.g. via the **File/ Disconnect from database** menu item).

The connection slot still exists, but it is not actually used.

As a consequence, if access to the database is based on the principle of floating users, the number of effective concurrent users is decremented by 1.

AssetCenter solves this problem by searching at regular intervals for unused connection slots, and using them for a new connection. Here is how AssetCenter determines if a connection slot is obsolete:

1. AssetCenter searches for the date the slot was last modified.
2. If the time at which the slot was last modified is twice as old as the slot update frequency, AssetCenter considers that the slot is inactive and therefore can be reused.

An administrator using the "Admin" login can also fix this problem by deleting obsolete connection slots. Because connection slots are refreshed regularly during the database connection, it is easy to delete obsolete records from the table of connection slots:

1. Use a query filter to display all records in the table of connection slots whose **Modified on** field indicates a date earlier than a given date.
2. Select those slots and delete them.

Disconnecting inactive users

AssetCenter allows you to disconnect floating users automatically according to a given timeout value.

This is defined via the **Tools/ Database options** menu item.

1. The **Automatic disconnection** in the "Access control" section option allows you to specify if you want to activate the automatic disconnection procedure.
2. If you decide to activate the automatic disconnection procedure, then you need to define a value in the **Slot timeout in seconds** option of the "Access control" section.

Warning: In order for your modifications to be recognized, you need to disconnect, then reconnect.

Note: If you use a AssetCenter version 3.01 or earlier, the automatic disconnection is applied to the ensemble of users.

Managing passwords

This section explains how to manage passwords with AssetCenter.

You will find information on the following topics:

- Password for the Admin login.
- Modifying the password for the Admin login.
- User password.
- Lost passwords.

Password for the Admin login

The record of the "Admin" login is very important:

1. For the first installation of AssetCenter, this is the only login name available for accessing the AssetCenter database for all administrative operations.
2. You can subsequently assign administrative rights to other records in the table of departments and employees. However, the "Admin" login record cannot be deleted; it is your only solution if you can no longer connect under another **Login** name that also has administrative rights.

The password associated with the "Admin" login is therefore critical, because it gives all rights to the AssetCenter database.

Note: Warning: Do not forget the passwords for the "Admin" login, or you will never be able to administer the database if the other records in the table of departments and employees with administrative rights are damaged.

Modifying the password for the Admin login

You can modify the password for the "Admin" login record by opening the database under this login name and using the **Tools/ Change password** menu item.

User password

Modifications performed by the user

Every AssetCenter user can modify their password by opening the database under their Login name and then selecting the **Tools/ Change password** menu item.

Modifications performed by the administrator

The administrator may modify user passwords in the employee's **Profile** tab. This overwrites the old password.

Lost passwords

User password

AssetCenter does not display passwords in a readable format. If an AssetCenter user loses or forgets their password, an administrator must create a new one in the employee's **Profile** tab. This erases the old password.

"Admin" login password

If you have forgotten your administrator password despite the warnings above, there is no way to recover it in AssetCenter. You therefore will need to contact Peregrine System support.

7 Writing queries in AQL

CHAPTER

This chapter explains how to write queries in AQL:

Presentation

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

- Queries in AssetCenter
- AQL

Queries in AssetCenter

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

Within the AssetCenter suite, 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 Crystal Reports.
- Create wizards.
- When you use the AssetCenter APIs and/or AssetCenter WebKit.
- If AssetCenter is used as a DDE server.

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

AssetCenter 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.

Warning: 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.

AQL

AQL ("Advanced Query Language") is the querying language used by AssetCenter to access the AssetCenter 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 knowledge of databases before using AQL directly.

Usefulness of AQL

AQL is better suited to querying the AssetCenter database than SQL for the following reasons:

Independence from database

The various database engines supported by AssetCenter all use differing versions of SQL, which 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, AQL uses the same set of functions, regardless of the database engine used.

Thus, the "Substring" function in AQL is equivalent to "Substr" under Oracle for WorkGroupsSQL 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 in the utilization of sorts and indexes. For example, when searching assets (sorted by brand, using the indexes) you would write the following in AQL:

```
 SELECT FIRST_ROWS Brand FROM amAsset ORDER BY
Brand
```

The SQL code generated will differ according to the DBMS used and will be optimized for this. Thus, under Oracle for WorkGroups 7.3, the SQL code will be:

```
SELECT /*+ FIRST_ROWS INDEX_ASC(a1 Ast_Brand) */
a1.Brand FROM amAsset a1 WHERE a1.Brand >= CHR(0)
```

Under SYBASE 11, the SQL code will be:

```
SELECT a1.Brand FROM amAsset a1(Index Ast_Brand)
```

Under Microsoft SQL Server 6.5 the code will be:

```
SELECT a1.Brand FROM amAsset a1(Index=Ast_Brand)
```

Simplified access to the AssetCenter database

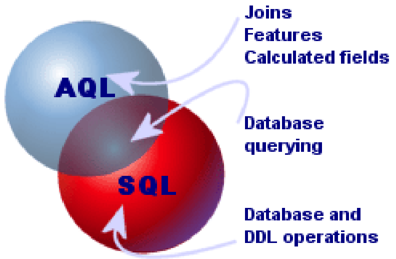
AQL simplifies the management of links and joins. This greatly facilitates access to the database when writing queries as compared to using SQL directly.

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.

How AQL compares to SQL

Figure 7.1. Comparison of AQL and SQL



The above diagram illustrates how AQL compares to SQL:

- AQL is compatible with SQL database querying statements ("SELECT").
- AQL does not have an equivalent for SQL statements, which write to the database ("INSERT", "UPDATE", "DELETE") or DDL statements ("Data Definition Language").

Warning: You should never write to the AssetCenter database directly using SQL statements.

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

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 AssetCenter database that have an effect on the optimal design of queries.

The sections entitled "AQL Syntax" and "AQL Functions" complement this section.

You will find the following sections:

- Presentation of AQL joins.
- Reason for and usefulness of primary key 0 records.
- Usage of NULL.
- Self
- CurrentUser
- System itemized-lists
- Hierarchic tables
- Simplified AQL notation

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

Presentation of AQL joins

Definition

A join consolidates multiple data tables in a single query.

AQL joins

AssetCenter'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 AssetCenter database.

Example

The following query, written in AQL, returns for each Asset Tag, the name of its user and its supervisor:

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

Here is the same query written in Oracle for WorkGroups SQL:

```
SELECT B1.AssetTag, U2.Name, R3.Name FROM amAsset
 B1, amEmplDept U2, amEmplDept R3 WHERE B1.lUserId
 = U2.lEmplDeptId AND B1.lSupervId = R3.lEmplDeptId
```

The two joins between the table of assets and the table of departments and employees are automatically handled under AQL. Using the AssetCenter's graphic query editor, you just have to click the fields of a selected table or linked table in a tree-structured list in order to create the corresponding AQL code.

Note: On all systems except Oracle, the number of outer joins is limited to 1.

```
useSQL92Join=1
```

Reason for and usefulness of primary key 0 records

Primary key "0" records

The AssetCenter data module has certain specificities:

- The primary and foreign keys of each table are numeric (32-bit integer).
- 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 Asset Tag, the name of its user and its supervisor:

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

An asset 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 table of departments and employees.

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 asset, 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 amAsse
```

If the generated SQL code used the outer joins of the DBMS, the SQL code generated for Sybase SQL Server would be:

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

This code is not supported by Sybase SQL Server 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. AssetCenter thus generates SQL code without outer joins:

```
SELECT l.name FROM amAsset 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 amAsset
```

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 in order 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 manage outer joins at DBMS level, use the "=*" and "*=" AQL operators.

Usage of NULL

AssetCenter uses the NULL value of the DBMS in two cases only:

- For an empty "text" type field.
- For a non populated "date" or "date+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+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" type 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 "lLocaId=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 AssetCenter 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 amEmplDept
```

Is equivalent to:

```
SELECT Name + "," + FirstName + "(" + Phone + ")"
 FROM amEmplDept
```

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 manually as it is not offered by the query editor.

Used as "expression"

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

```
SELECT lAstId FROM amAsset WHERE User = CurrentUser
```

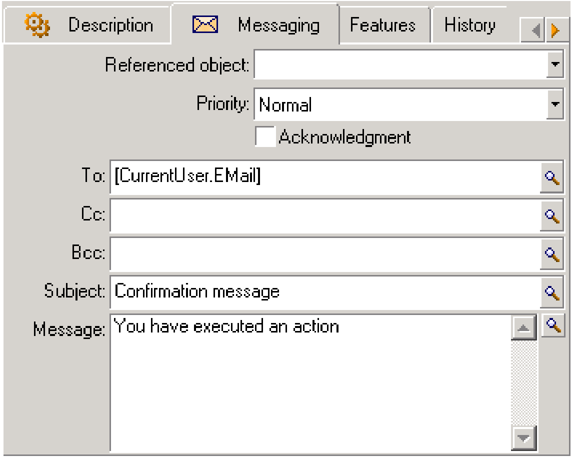
Used as "link"

"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 just need to populate the detail of the action as follows:

Figure 7.2. "CurrentUser" function used as variable in an action



Referenced object:

Priority: Normal

□ Acknowledgment

To: [CurrentUser.EMail]

Cc:

Bcc:

Subject: Confirmation message

Message: You have executed an 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 AssetCenter Database Administrator, or the database.txt file, which describes the structure of the database.

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.

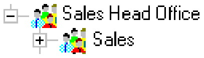
```
FullName = '/Milwaukee/Water St. Site/Building
A/5th floor/'
```

"sLvl" fields

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

Root level is level 0.

Figure 7.3. Example in the table of departments and employees



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 string

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 amAsset
```

Is equivalent to the query:

```
SELECT user.self FROM amAsset
```

That is itself equivalent to:

```
SELECT User.Name + "," + User.FirstName + "(" +
User.Phone + ")" FROM amAsset
```

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 and prefix this by "fv_".

Example: The following query searches the values of the feature with SQL name "XXX" for the table of assets:

```
SELECT fv_XXX FROM amAsset
```

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 and prefix this by "cf_".

Sorts and indexes

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

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

Note: Note: The choice between these two different methods is not available under SQL Anywhere. The database engine automatically selects the most suitable method.

Example

In the case of the following query:

```
SELECT lAstId, Brand FROM amAsset 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 in which you force the utilization of the indexes depends on the way in which you create the query.

Via the Configure list menu

You can configure the data-access type for each list in AssetCenter. 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 to display the shortcut menu.
3. Select **Configure list**.
4. In the **Columns and sort** tab, check the **Force indexes** box in order to use the indexes indicated in the query and display the results as and when they are generated. Uncheck this box in order to select the other type of access.

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 characteristic SqlName.amFeature.seDataType), there is a possibility that the sort is not valid.

Sort order

The sort order depends on:

- The database engine.
- The use of indexes.

Under Oracle for WorkGroups

With indexes forced

- NULL records do not appear.
- The sort is performed according to the value of the ASCII codes thus differentiating between upper and lower case characters (binary sort).

Without indexes forced

- NULL records appear.
- Oracle for WorkGroups is not case sensitive.

Example

Table 7.1. Example of sort order without indexes forced

Starting list	A B C D a b NULL NULL
List with indexes forced	A B C D a b
List without indexes forced	NULL NULL A a B b C D

Under Microsoft SQL Server or Sybase SQL Server

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

Under Sybase SQL Anywhere

Under Sybase SQL Anywhere, the indexes cannot be forced via an AQL query.

The database engine determines the optimal method used to access the data and to sort it.

Precautions

With complex queries it often difficult to determine whether it is more "advantageous" 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).

The query editor

AssetCenter 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.

This sections details the functioning of the query editor:

- Principle
- Accessing the query editor
- Creating a query using the query editor
- Fields used in a query
- Writing an expression
- Constants

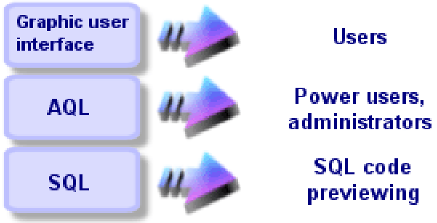
Overview

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.

Whether you use the graphic method or prefer to write directly in AQL (both approaches are frequently combined), you get to see a real-time transcription of your query in SQL. However, you cannot write your queries directly in SQL.

Figure 7.4. Intended uses of the query editor



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

You can access the query editor in different ways:

- 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:
  - Either directly via the window displayed by the **Tools/ Queries** menu item.
  - Or by when using a "query filter" when displaying the main table of the query.
- Via the numerous functions of AssetCenter that call on queries: Access restrictions, query filters, list configuration, etc.
- Via external programs: AssetCenter Export, etc.

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, etc.), you only have to define the WHERE clause of the query. The other components of the query (starting table, fields, etc.) 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 **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 amAsset WHERE Brand='Compaq'
```

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

```
Brand='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.

Creating a query using the query editor

To create a query using the query editor, select the **Tools/ Queries** menu item. The window that is displayed 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.

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 able to be accessible to other users, uncheck the **Not shared** option (SQL name: bPrivate).

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 Create to be able to access the detail tabs of the query.

Step 2: Define the filter conditions in the Filter (WHERE clause) tab.

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

You can define one or more filter conditions.

To define a filter condition:

1. From a start table, select a field, constant or expression (**Field 1**), that you will compare with a field, constant or expression (**Field 2**).
2. Confirm the filter condition by transferring it to the lower part of the screen using the ⬇ button.
3. Confirm the query by clicking Modify in the query detail.

To define several filter conditions linked by the AND and OR logical operators:

1. Create a first filter condition as indicated above.
2. Define the other filter conditions and confirm them using the ⬇ AND or ⬇ OR buttons.
3. Confirm the query by clicking Modify in the query detail.

Note: In order to make a modification to the selected conditions, click the button to delete the contents of the window, or else modify the AQL code directly.

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

Step 3: Preview the execution of the query

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: AssetCenter 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

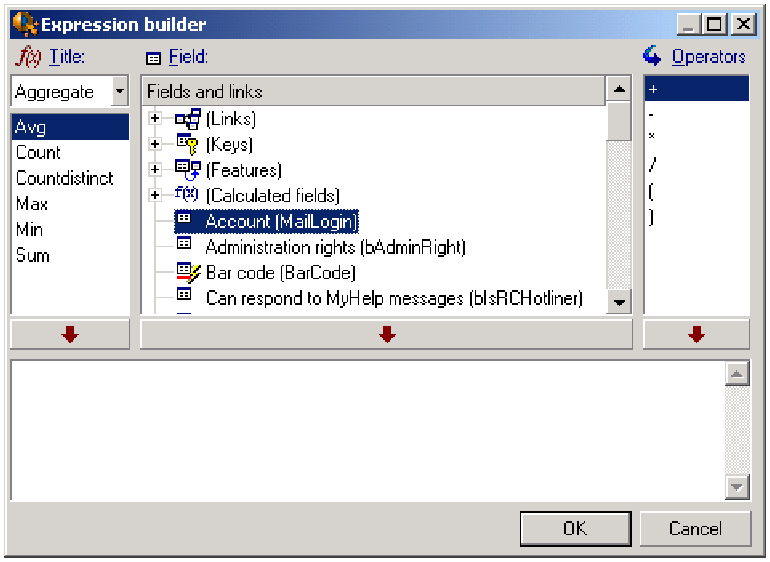
Expressions f(x) enable you to perform calculation operations in your query. For example, you can use the "Count" function to count the number of resultant records of a query.

To write an expression, you can either:

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

To use the expression builder, click the 🔍 button adjacent to the edit zone of the **Filter (WHERE clause)** tab of the query detail.

Figure 7.5. Expression builder



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.

Constants

Constants K are fixed values that you assign to selection criteria. For example, if you are searching all assets with brand "3Com", you assign the constant value "3Com" to the **Brand** field (SQL name: Brand) in the table of assets.

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.

AQL syntax

This section details the syntax of AQL:

- Conventions
- Syntax of queries
- FROM clause- Tables involved in a query
- Elements of a query
- WHERE clause
- GROUP BY clause
- HAVING clause
- ORDER BY clause

Note: 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, please consult the appropriate reference documentation.

Conventions

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

Table 7.2. 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>]

)

Note: Attention: Sub-queries must be contained with parentheses.

Example of utilization:

```
SELECT Self FROM amAsset WHERE dPrice >= (SELECT
Max(dPrice)/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>]

[GROUP BY <clause>]

[HAVING <clause>]...]

[ORDER BY <clause>]

FROM clause- Tables involved in a query

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

Syntax

AQL authorizes the utilization of aliases for table names.

The FROM clause takes the following form:

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

Examples:

```
FROM amAsset
FROM amAsset a, amLocation l
```

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, amLocation
```

Examples of utilization

The following queries are equivalent:

```
SELECT AssetTag FROM amAsset
SELECT AssetTag FROM amAsset a
SELECT a.AssetTag FROM amAsset a
SELECT amAsset.AssetTag FROM amAsset a
SELECT amAsset.AssetTag
SELECT amAsset:AssetTag
```

Number of tables in a query

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

Example:

- Oracle for WorkGroups: You can use as many tables as you like.

- Microsoft SQL Server or Sybase SQL Server: You are limited to 16 tables in a query.

Note: Important note: 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.

Elements of a query

Fields and links

Queries involve fields and links in the AssetCenter 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 table of assets:

```
Brand
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 AssetCenter, 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 (SQL name: TableName) in query detail, etc.). 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'

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#

#1999-01-01 01:00:03#

Expressions

Expressions are formed using:

- Constants
- Fields
- Functions
- Sub-queries

You can combine these elements with operators and parentheses in order 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:

Table 7.3. 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 assets 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.

Table 7.4. 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 in which AQL handles links, the use of this operator is limited
*=	Left outer join. Because of the way in which 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 Anywhere, SQL Server and Sybase support 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. DB2 does not support use of the LIKE X operator, if X includes a SQL column name. Only constants are supported for this operator. For example, the following query is not correct for DB2: SELECT COL1, COL2 FROM TABLE1 WHERE COL1 LIKE COL2
IS NULL IS NOT NULL	Tests whether the value of a field is "NULL" or not. AssetCenter only authorizes the "NULL" value for empty text fields and for **Date** or **Date+Time** fields that are not populated.

Note: SQL Anywhere is not able to process "LIKE X" clauses when X contains more than 128 characters. If X is larger than 128 characters applying the query provokes an ODBC error message. This problem can, for example, occur when displaying lists in tree view since this operation uses a "LIKE" clause on a "FullName" field.

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 query provides the list of assets whose brand is used at the Milwaukee site:

```
SELECT lAstId, Brand FROM amAsset WHERE Brand =
ANY (SELECT Brand FROM amAsset WHERE
location.fullName='/Milwaukee')
```

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: Important note: Certain DBMSs limit the number of expressions in a given SELECT statement.

WHERE clause

The AQL "WHERE" clause is equivalent to the "WHERE" clause in SQL.

It specifies the search conditions.

WHERE <Search conditions>

The search conditions specify the items to be extracted from the database and can be expressed in the WHERE or HAVING clauses.

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 "GROUP BY" clause in SQL.

GROUP BY <expression without aggregates>

[, <expression without aggregates>]...

"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.

Examples of queries with and without the "GROUP BY" clause:

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

```
SELECT Count(Brand) FROM amAsset
```

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

```
SELECT Brand, count(lAstId) FROM amAsset GROUP BY
 Brand
```

HAVING clause

The AQL "HAVING" clause is equivalent to the SQL "HAVING" clause.

HAVING <Search conditions>

The "HAVING" clause 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.

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 Brand, count(lAstId) FROM amAsset GROUP BY
 Brand HAVING Brand > 'B'
```

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

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

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 Brand, count(lAstId) FROM amAsset GROUP BY
 Brand HAVING count(Brand) > 1
```

ORDER BY clause

The AQL "ORDER BY" clause is equivalent to the SQL "ORDER BY" clause.

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

Items can be sorted:

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

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.

Aggregate-type AQL functions

Table 7.5. Aggregate-type AQL 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 in a "number", "string" or "date" type column. If the column does not have any records, returns "0" ("number" type column), "empty string" ("string" type column), or "empty date" ("date" type column).
Min(<column>)	Returns the minimum value in a "number", "string" or "date" type column. If the column does not have any records, returns "0" ("number" type column), "empty string" ("string" 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.

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

String-type AQL functions

Table 7.6. String-type AQL 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".
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>.
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).
Upper(<String>)	Returns the <string> in uppercase.

Date-type AQL functions

Table 7.7. Date-type AQL functions

Function	Description
Year(<date>)	Returns the number representing the year for a "date" or "date and time type field" (e.g: 2000).
Month(<date>)	Returns the number of the month for a "date" or "date and time type field" (1=January, ?, 12=December).
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).
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 Sybase or Microsoft SQL Server is (1=Sunday, 2=Monday, ?, 7=Saturday). The default configuration under Oracle is (1=Monday, ?, 7=Sunday).
Hour(<hour>)	Returns the number of the hour in the day for a "time" or "date and time" type field (0-23).
Minute(<hour>)	Returns the number of minutes for a "time" or "date and time" type field (0-59).
second(<hour>)	Returns the number of seconds for a "time" or "date and time" type field (0-59).
Getdate()	Returns the server's current system date.
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.
DaysDiff(<date1>, <date2>)	Number of days between the dates date1 and date2 ("floating point" number with decimals)
HoursDiff(<date1>, <date2>)	Number of hours between the dates date1 and date2 ("floating point" number with decimals)
MinutesDiff(<date1>, <date2>)	Number of minutes between the dates date1 and date2 ("floating point" number with decimals)
SecondsDiff(<date1>, <date2>)	Number of seconds 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.

Function	Description
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.

Examples:

Table 7.8. Examples of Date-type AQL functions

Description	AssetCenter 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

Table 7.9. Numeric-type AQL functions

Function	Description
Abs(<Number>)	Returns the absolute value of a "number".
Ceil(<Number>)	Returns the smallest integer greater or equal to a "number".
Floor(<Number>)	Returns largest integer less than or equal to a "number".
Mod(<a>, <b>)	Returns the remainder of the division of "a" by "b" (a = qb + r, with q integer and 0 £ r < q).
Round(<a>, <n>)	Rounds "a" to "n" decimal places.
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

Table 7.10. Test type AQL functions

Function	Description
IsNull(<a>, <b>)	If "a" is "Null", replaces "a" by "b". The types of "a" and "b" must be compatible.

Examples of queries

The following examples of queries each 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 AssetCenter Export. 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 Brand='Compaq'
```

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

```
Brand='Compaq'
```

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

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.

To compare a field in the main table with a value

Example: All "Compaq" brand assets.

```
SELECT Self FROM amAsset WHERE Brand = 'Compaq'
```

To compare a link in the main table with another link

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

```
SELECT Self FROM amAsset 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 assets that have the same location name as their parent.

```
SELECT Self FROM amAsset 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

Example: All work orders carries out between January 1, 1995 and December 31, 1995.

```
SELECT Self FROM amServiceCall WHERE (dtFirstCall
 >= #95/01/01 00:00:00#) AND (dtFirstCall <=
#95/12/31 00:00:00#)
```

Query concerning a feature

Example: All assets whose feature with SQL name "Size" showing a value greater than or equal to 150 cm.

```
SELECT Self FROM amAsset 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 in order 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 than an empty string is represented by two single quotes '.

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

To search for the absence of a link

Case of a 1 link

Example: All assets that have not been assigned to a user. Note that the absence of a link is denoted by "0".

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

Case of n links

Example: All categories with no associated assets:

```
SELECT Self FROM amCategory WHERE 0 = (SELECT
COUNT(a2.lAstId) FROM amAsset a2 WHERE a2.lCategId
 = lCategId)
```

This query scans the table of categories, takes each category on after the other and compares with 0 the number of assets belonging to this category.

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

Example: All assets without parent asset or sub-asset:

```
SELECT Self FROM amAsset WHERE (0 = (SELECT
COUNT(a.lAstId) FROM amAsset a WHERE a.lParentId
= lAstId)) AND (Parent = 0)
```

This query performs:

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

Another example

All assets without "Hard Drive" category sub-component:

```
SELECT self FROM amAsset p WHERE NOT ( EXISTS
(SELECT lAstId FROM amAsset WHERE (FullName LIKE
(p.FullName + '%/')) AND (Category.Name = 'Hard
Drive')))
```

Query with alias

Example: All employees having taken the 'Peregrine' training program and the a 'Database' training program.

Start table: The table of departments and employees.

The query is as follows:

```
SELECT Self FROM amEmplDept WHERE (Training_1.Title
 = 'Peregrine') AND (Training_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 (Training.Title
 = 'Peregrine') AND (Training.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 (Training.Title
 = 'Peregrine') OR (Training.Title = 'Database')
```

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

CHAPTER

8 Editing forms

This section explains how to design forms with AssetCenter.

Use the File/ Forms menu item to display the list of forms.

Definition of a form

A form is a document model that lets you print data.

Unlike Seagate Crystal reports, forms are created directly in AssetCenter.

Creating forms

Use the File/ Forms menu item to display the list of forms.

Basic information

1 Enter the name of the form.
2 Select the type of form: (list or detail)

Both types can contain text and predefined images.

Differences between the two types:

- List: enables you to print a list of records such as it is displayed in the active list (according to the columns the list contains and the filters applied).
- Detail: enables you to print fields from a record detail (example: the detail of an asset), and linked record lists (example: the components of this asset).

3 Select the main table for the form.

Note: Warning: The Table field (SQL name: TableName) lets AssetCenter propose only those forms relevant to the given list.

Editing forms and objects

To edit a form you define objects and position them on the page.

To insert a new object on the page:

1 Select the Form tab.

2 Click the icon of the object that is on the left-hand side of the page.

Table 8.1. Editing forms and objects - icons

Icon	Function
↖	**To select an object in the form in order to modify it, for example.**
A	To add fixed text and variables independent of the records being printed (current date, for example).
▨	To add an image.
f(x)	To insert a formula containing field values and fixed text strings.
▦	To insert a list of records. This tools enables you to position the list on the page. For detail forms, it also enables you to define the linked table containing the records and list of fields to be printed.

3 Position the mouse cursor on this page.

4 Click the left mouse button.

5 Trace a frame with the mouse: This frame defines the space reserved for the object.

6 Double-click the object's position. A window describing the object's properties is displayed.

7 Define the properties of the object.

8 Click Modify.

You can insert the following objects in a form:

Fixed text

This concerns text that is independent of the records being printed. You can combine any kind of character, as well as variables:

$D: Date printed.

$U: Name of the AssetCenter user who is printing the form

$C: Page number.

$N: Total number of pages printed.

Warning: Do not put quotes around the text.

Example:

```
Document printed on $D on $U
```

Formulas

Formulas are only available in "detail" type forms.

Formulas bring together:

- field values from the AssetCenter database.
- fixed text surrounded by quotes.

Example:

```
"Asset:" CodInt " / " Brand
```

Formulas do not permit you to perform calculations.

Lists

- "List" type forms: You can only include one list. This list will be replaced by the list in the active window when you select the **File/Print** menu item.
- "Detail" type forms: The number of lists is not limited. Lists display all records related to the current record. For example: All the components of an asset.

Images

You can insert images (logos, etc.).

Note: When you draw up a "list" type form you cannot select the fields to print. AssetCenter prints the fields that appear in form of columns in the lists.

Properties of object in forms

Position and size

To modify the position and size of an object, simply drag the object or its edges.

Note: You can move and re-size several objects at once by selecting them simultaneously. If you draw a selection frame around several objects with the mouse, all objects in the frame will be selected; or select the objects one by one, while holding down the CTRL key.

Properties

If you double-click an object, or use the **Internal forms/ Properties** menu item, a list of the selected object's properties is displayed.

The property list has two columns: The first contains property names and the second lets you edit the property values.

To modify a property, click in the second column with the left mouse button.

Simple properties may be edited directly (text, formulae, lists, background colors, text colors, text alignment, object alignment.) Complex properties (font for the text or formula, frame, list contents, image) display additional windows.

Text

Enter your text directly.

Formula

To help you in creating formulas, you can display a drop down list with a tree that displays the fields that are accessible and compatible with the type of form selected. Clicking a node in the tree replaces the current selection with the field you selected in the list. You can insert text between fields as long as you include quotes.

Background color, text color

You may choose these colors from 16 available colors.

Text alignment

This determines how text is aligned within the frame. A drop-down list displays the different kinds of alignment (Centered, Left aligned, Right aligned.)

Page alignment

This determines the text's horizontal alignment in the page. A pop-up window lets you select the alignment:

- **Left aligned**.
- **Right aligned**.
- **Centered**.
- **Relative alignment**: In this case, the object keeps the position you defined in the form creation screen.

Font

To select the character set of the object and its size, click 🔍 next to cell being edited.

Frame

To add a border around objects, click the 🔍 button in the edit cell: A configuration screen for the border is displayed.

The **3D** style gives an embossed effect to the frame.

For borders without the 3D style, you can select which borders will be displayed, their color and thickness.

Image

To insert an image:

1. Click the button in the edit cell.

2 Select the graphic file in the "Select image" screen.

Link in list

To select a list to be displayed in the form, use the drop down list. Click the list you want (for example, the list of assets used by an employee).

List contents

Note: You can only configure the contents of the list when editing a detail form.

To define the contents of a list, click the button in the edit cell. A configuration window is displayed. All columns in the list are shown.

This screen lets you define:

- the title of each column,
- the formula used to define the contents of the column,
- the size of the column, (percentage used by column)
- the alignment for the title and contents of each column,
- the alignment of each title or contents of each column,
- the vertical and horizontal separators.

To remove a column from the list, use the "Delete" key.

To insert a column in the list, edit the last line of the list.

Each cell can edited using the same principles as for the list of properties.

Design grid

The design grid is made up of vertical and horizontal lines that cover the background of the screen.

The **Form/ Grid** contextual menu item allows you to:

- Show and hide the design grid.

- Define the spacing between line in the grid.

Only the points of intersection between horizontal and vertical lines are shown. The spacing between lines determines the precision with which you can position objects on the page.

Form page setup

The **Form/ Page setup** contextual menu allows you to define:

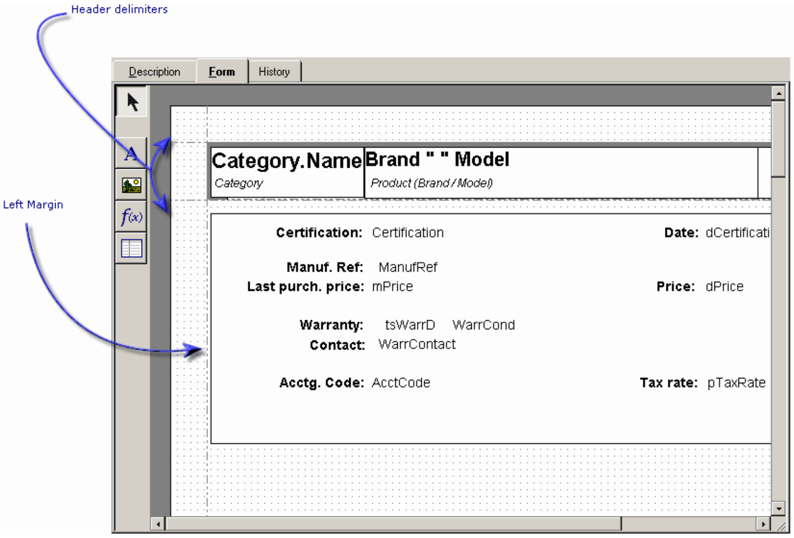
- The page format
- **Portrait** or **Landscape** layout
- Document margins
- Header and footer

Note: Document margins, footer and header zones can be directly modified in the edit zone. Simply drag the margin marker and zone limits for the header and footer, which are shown in dotted lines.

To insert text in page headers and footers:

1. Select the **Form/ Page setup** menu item.
2. Check the **Header** and **Footer** boxes.
3. Click OK.
4. Go to the header or footer zone in the form (header and footer zones are shown with horizontal dotted lines).
5. Insert objects here or move objects from the main page zone.
6. Confirm your modifications by clicking Modify.

Figure 8.1. Form header



Category.Name Brand " " Model
Category Product (Brand / Model)

Certification: Certification Date: dCertificati
Manuf. Ref: ManufRef
Last purch. price: mPrice Price: dPrice
Warranty: tsWarrD WarrCond
Contact: WarrContact
Acctg. Code: AcctCode Tax rate: pTaxRate

Note: You cannot move objects from the header and footer zones to the main page zone.

How to easily create regular reports

To produce reports you need regularly, we recommend that you:

1. create a "view" with the appropriate parameters.
2. associate this view with a form.

The view lets you define:

- The sort criteria.
- The filter to apply and the filter values.
- The list of visible columns.

The form lets you organize the report page layout.

To print the report:

1 Display the view you created beforehand (**Tools/ Views** menu item.)

2 Print from the displayed view (**File/ Print** menu item.) Make sure you select the correct "Type" of report and the appropriate "Form".

9 Editing reports

CHAPTER

This chapter explains how to print reports with AssetCenter.

Use the File/ Reports menu item to display the list of reports

Operation and installation of the report generator

Overview

AssetCenter makes use of Crystal Reports reporting software. These reports have the file extension **.rpt**.

You do not need to have Crystal Reports to print existing reports

A limited version of Crystal Reports is installed with AssetCenter if you check the appropriate option during the installation.

This limited version is sufficient to preview and print existing reports with the current data from the AssetCenter database.

You need to have Crystal Reports to modify existing reports or to create new ones

AssetCenter obviously cannot let you create these reports directly.

For this you need to install Crystal Reports version 8.

Installing, configuring and inserting Crystal reports in your database

Please refer to "Installation and Upgrade Guide", chapter "Installing AssetCenter for the first time", section "Installing the reporting program".

Detail of a report

Use the **File/ Reports** menu item to display the list of reports.

A report detail in AssetCenter is made up of the following information:

File

You cannot edit this field directly. It indicates the name of the report file (with its extension and the relative path of its folder) that was imported using the Import....

The following buttons let you work with reports:

- Import...: This button in the report detail enables you to import (the first time to create the report, following times to modify the report) an external report. External reports have the **.rpt** file extension. Importing an external report updates the **File** field (SQL name: FileName) in the AssetCenter report detail.
- Export...: This button in the report detail enables you to create an **.rpt** file from a report contained in the AssetCenter database. By default, the dialog box that opens proposes the name of the file contained in the **File** field. You can modify this. Doing this allows you to modify a report using the external reporting program.

- Preview...: This button, accessible via the **File/ Print** menu item, lets you preview the report on-screen before printing it.
- Print: This button, accessible via the **File/ Print** menu item, lets you print the report.

Note: When you press the or buttons, AssetCenter creates a temporary file from the report in the database. This file is processed by the Crystal Reports print engine. The temporary file is erased immediately afterwards. The data displayed or printed is the data currently in the database.

How to modify a Seagate Crystal report

In order to modify a report contained in the AssetCenter database, you need to have Crystal Reports.

Use the following procedure:

1. Display the detail of the report using the **File/ Reports** menu item.
2. Click the Export... button to create a **.rpt** file.
3. Modify the **.rpt** report using Crystal Reports and save it.
4. Display the report detail again using the **File/ Reports** menu item.
5. Import the **.rpt** file to update it and modify the record.

Crystal Reports statistics

To display Crystal reports that are automatically updated, use the **Tools/ Crystal Reports statistics** menu item.

You can display the same reports as when using the **File/ Reports** menu item.

Nature

Indicate the nature of the report to be displayed. The field to the right of this field enables you to select a given report. The reports that are available depend on the "Nature" you select.

⟳ button

- Click this button to refresh the report.
- Right-click this button to define the frequency of automatic refreshing of reports.

🗗 button

Modifies the zoom factor (3 levels).

Creating a detail report

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

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 Crystal Reports

To obtain a detail report, follow the procedure below (example taken under Crystal Reports Professional 5.0):

1 Use the **Insert/ Formula Field** menu item to create a formula field. Its name must respect the following syntax:

```
<SQL name of the table for which the report is
cotextuall>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 "lAstId", which is different from "amAssetId".

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 AssetCenter database.

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 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.lAstId} = {@amAssetId}
   ```

Using the above procedure, AssetCenter 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 **File/ Reports** menu item.
2. Create a new report.
3. Import the Crystal Reports file (**.rpt** extension) by clicking the 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 Crystal Report.

10 Read-only access to the database using the AssetCenter ODBC driver

CHAPTER

This chapter explains how to use external tools to access (read-only) the AssetCenter database via the ODBC driver developed for AssetCenter.

Note: The AssetCenter ODBC driver only supports read-only access to the database.

Overview of access to the AssetCenter database

Installing the ODBC driver

The AssetCenter installation program will install the ODBC driver if you:

- Specifically select this package during setup.
- Or if the driver is required by other installed packages.

The ODBC driver is called Peregrine AssetCenter Driver. **Adbc.dll** is copied into the Windows "system32" folder.

Note: The ODBC driver is independent of the language version of AssetCenter and the DBMS used.

When to use the ODBC driver

Using this driver is recommended when designing reports for the database using external tools such as Crystal Reports.

Note: You are not obliged to use this driver. You can access the AssetCenter database directly if your reporting program supports the relevant DBMS directly.

Advantages of using the ODBC driver:

Table 10.1. Advantages of using the ODBC driver

	Using the ODBC driver	Without the ODBC driver
Security when accessing the AssetCenter database	When using reports, an AssetCenter login and password are required to access the database. User profiles associated with the login are respected.	When using reports, the database access parameters required by the DBMS are requested. These are not linked with the AssetCenter user profiles.
DBMS connection parameters	You don't need to know the connection parameters required by the DBMS in order to access the database.	You need to know the connection parameters required by the DBMS in order to access the database.
Selecting the connection used to access the AssetCenter database.	The user of the report selects the appropriate connection.	The user directly accesses the database without using the AssetCenter connections.
Link between the DBMS engine and the report	The report is independent of the database engine used. When you change DBMS, you don't have to modify your reports.	The report depends on the database engine. When you change DBMS, you have to modify your reports.

Data items accessible via the ODBC driver

The ODBC driver enables you to view:

- Tables
- Standard fields
- Calculated fields
- Features

All these objects are identified by their SQL name.

Note: Links are not visible. You must reconstitute them by making the joins yourself.

Which ODBC connection to use

A standard ODBC connection is created when the ODBC driver is installed. It is called AssetCenter Databases. This connection cannot be modified or removed.

You can use two types of ODBC to connection to access the AssetCenter database:

- The standard AssetCenter Databases connection.
- A connection you have created yourself.

Usefulness of the standard AssetCenter Databases connection

By using this connection, you avoid having to create your own. In this way, you don't have to use the ODBC administrator. The AssetCenter connection to be used is selected when creating and using the report. The standard connection dialog box under AssetCenter is used for this.

How to create your own ODBC connections

1. Launch the ODBC administrator.
2. Create a new connection by selecting the Peregrine AssetCenter Driver.
3. Proceed as usual to create the ODBC connection.

Example: Creating a report under Crystal Reports with the ODBC driver

- Launch Crystal Reports.
- Open a new report.
- Indicate than the report concerns "SQL/ODBC" data.
- Select the AssetCenter Databases ODBC connection.
- The standard AssetCenter connection dialog box is displayed.
- Select the appropriate AssetCenter connection. Enter the login to be used to create the report and the appropriate password.
- Create the report in the usual fashion.

11 Defining actions

CHAPTER

This chapter explains how to define actions with AssetCenter.

Use the Tools/ Actions/ Edit menu item to create actions.

You can execute actions via the Tools/ Actions menu item or the contextual list of "Actions" in the toolbar.

Definition of an action

An action enables you to automate, either completely or partially, the tasks performed on an AssetCenter database.

There are several types of actions:

- Executable
- Messaging
- Script: modifying an object in the AssetCenter database
- Wizard
- Printing

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

AssetCenter 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/ Active modules menu item.

Functional domains are used to create and classify the information displayed in the Functions and favorites pane. Thus, once you select a functional domain for an action, the action will appear in the Functions and favorites pane under the heading of that functional domain.

Note: The contents of the Functions and favorites pane is reorganized and modified according to the context. If your action is contextual (it can't be executed unless a specific screen is open, for example), then it will not appear in the Functions and favorites pane unless the current context corresponds to its definitions (if that specific screen is currently displayed, for example).

To define a functional domain:

1. Click **New**.
2. Select the **Administration/ Functional domains** menu item.
3. Assign a **Name** to your functional domain. This name is the one that will appear in the Functions and favorites pane. By default,

AssetCenter 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**.

Creating an action

This section describes how to create an action:

- Types of actions
- General method
- Populating the DDE tab
- Populating the Messaging tab

Types of actions

AssetCenter enables you to define several types of actions.

Note: AssetCenter only allows you to create Executable, Messaging or Printing type actions. DDE, Script and Wizard type actions are reserved and can be executed only.

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 it's extension is associated with an application in the file manager.

DDE actions

A **DDE** action sends a DDE request to a DDE server application (or DDE compliant application) capable of handling DDE requests.

DDE stands for "Dynamic Data Exchange"; it designates a method for dynamically exchanging information between programs. AssetCenter uses DDE have commands executed by another application.

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 AssetCenter's internal messaging system.
- Via an external VIM standard messaging system (Lotus Notes, Lotus cc:Mail, etc.).
- Via an external MAPI standard messaging system (Microsoft Exchange, Microsoft Outlook, etc.).
- Via an external SMTP standard messaging system.

Warning: 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, AssetCenter uses:

- The **Account** (SQL name: MailLogin) and **Password** (SQL name: MailPassword) fields of the **Messaging** tab of the detail of the employee who opened the AssetCenter database (table of departments and employees) 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 AssetCenter's internal messaging system, AssetCenter 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 AssetCenter user is the same as the Login.

Warning: The administrator must create a user with the name "Admin" and populate the Account, EMail and Password fields in order to use an external messaging system and make sure that AssetCenter Server functions correctly.

Script actions

Script actions enable you to perform any operation on an AssetCenter 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 AssetCenter database, e.g. 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 AssetCenter 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 AssetCenter 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 intended to guide you step by step though complex or recurrent tasks in AssetCenter. Wizards are designed via a dedicated programming language.

Note: Wizards are complex actions. They are documented in detail in the "Reference Manual": Administration and advanced use, "Presentation of wizards" and "Creating a wizards" chapters.

General method

To create an action:

1. Select **Tools/ Actions/ Edit.**
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**
- **DDE**
- **Messaging**
- **Script**
- **Wizard**
- **Printing**

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 AssetCenter is used as a DDE command server).

Note: If you do not populate the SQL name field, AssetCenter does so automatically by generating a standard SQL name.

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 Functions and Favorites pane.
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.

Note: The AssetCenter 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. Refer to the documentation of the executable for the list of services is 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 called application allows it, you can put several commands side by side.

  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.

Important 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, AssetCenter 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**.

Warning: 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 AssetCenter'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 AssetCenter database (in the departments and employees table).

Note: You cannot receive an acknowledgement for a message sent via the AssetCenter internal messaging system, or via a MAPI or SMTP messaging system.

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 AssetCenter's internal messaging system.
- MAPI: to force the utilization of a MAPI standard messaging system (Internet Mail, Microsoft Outlook, etc).
- VIM: to force the utilization of a VIM standard messaging system (Lotus Notes, etc.).
- 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 <AssetCenter 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 AssetCenter 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 AssetCenter'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 AssetCenter'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 AssetCenter 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

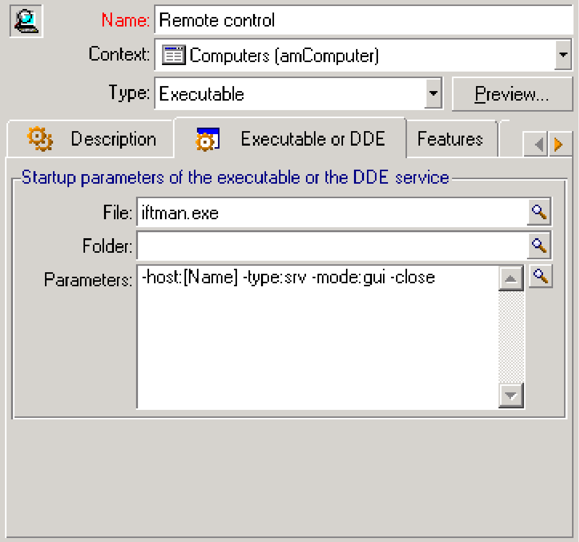
This section provides examples of AssetCenter actions:

- Example of Executable type action.
- Example of DDE type action.
- Example of Messaging type action.
- Example of a Script type action.

Example of an executable-type action.

The following screen describes a non-contextual action that launches AssetCenter Server and connects to a database **acdemo**:

Figure 11.1. Detail of an executable type action



Name: Remote control

Context: Computers (amComputer)

Type: Executable

Startup parameters of the executable or the DDE service

File: iftman.exe

Folder:

Parameters: -host:[Name] -type:srv -mode:gui -close

Example of a DDE-type action

There are numerous applications of **DDE** type actions:

- Inserting AssetCenter 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.
- Etc.

This section describes a simple **DDE** action.

Aim of the action

This action sends confirmation of a purchase request.

This action is triggered from the detail of a purchase request.

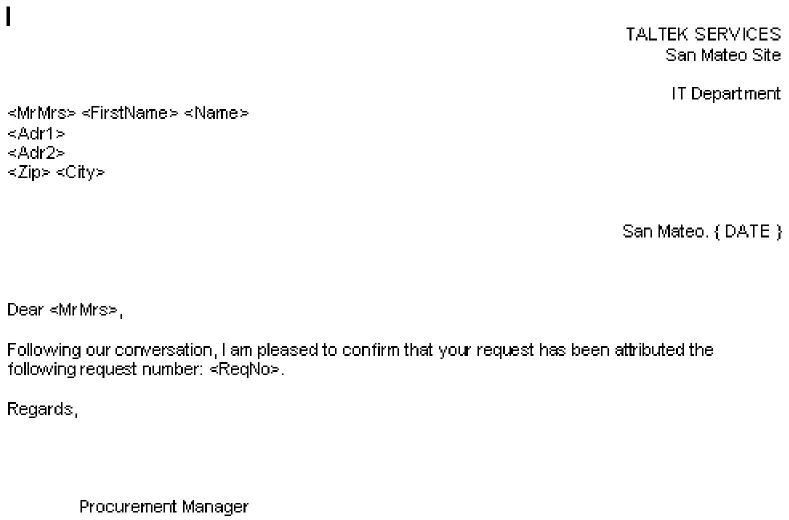
The action uses a DDE link between Microsoft Word 7 and AssetCenter. It inserts information on the request in a Word document (details of the person to contract and the request number) and prints it.

Preparations: Creating the letter under Word

You must first create the Word document **LetterTemplate.doc** that will be printed.

This letter will be structured as follows:

Figure 11.2. "LetterTemplate.Doc"



TALTEK SERVICES
San Mateo Site

IT Department

<MrMrs> <FirstName> <Name>
<Adr1>
<Adr2>
<Zip> <City>

San Mateo, { DATE }

Dear <MrMrs>,

Following our conversation, I am pleased to confirm that your request has been attributed the following request number: <ReqNo>.

Regards,

Procurement Manager

The **LetterTemplate.doc** document can be found in the installation folder of AssetCenter.

The corresponding Word template **Normal.dot** is attached to **LetterTemplate.doc**. It contains a macro, **mymacro.bas**:

```
Attribute VB_Name = "MyMacro"
Sub PrintLetterTemplate(MrMrs, FirstName, Name,
Adr1, Adr2, Zip, City, ReqNo)
'
' PrintLetterType Macro
'
   Application.WindowState = wdWindowStateMinimize
 'Run Winword in the back end
   Documents.Open ("LetterType.doc") 'Open letter
 pattern
    Documents("LetterType.doc").Activate

    Selection.Find.ClearFormatting 'Clear
parameters for Find function
    Selection.Find.Replacement.ClearFormatting
'Clear parameters for Replace function

    With Selection.Find
        .Text = "<MrMrs>"
        .Replacement.Text = MrMrs
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
'Execute replacement

    With Selection.Find
        .Text = "<FirstName>"
        .Replacement.Text = FirstName
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<Name>"
        .Replacement.Text = Name
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
```

```
        .Text = "<Adr1>"
        .Replacement.Text = Adr1
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<Adr2>"
        .Replacement.Text = Adr2
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<Zip>"
        .Replacement.Text = Zip
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<City>"
        .Replacement.Text = City
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    With Selection.Find
        .Text = "<ReqNo>"
        .Replacement.Text = ReqNo
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    ActiveDocument.PrintOut 'Print document
   ActiveWindow.Close (wdDoNotSaveChanges) 'Close
 document w/o updating
    MsgBox ("Your document is being printed.")
'Notify user
End Sub
```

Step 1: Creating the AssetCenter action

To create the AssetCenter action:

1. Open the table of actions (**Tools/ Actions/ Edit** menu item).
2. Click New to create a new action.
3. Populate the **Context** field (SQL name: ContextTable) in order for it to show the table of purchase requests.
4. Set the "Type" field to **DDE**.
5. Populate the **DDE** tab in order to define the DDE link between AssetCenter and Microsoft Word 7.

In the DDE tab:

1. The **Service** field (SQL name: DDEService) is set to "Winword".
2. The **Topic** field (SQL name: DDETopic) is set to "System".
3. The **Service start parameters** frame shows **Winword.exe** and its path.
4. The **Command** field (SQL name: DDECommand) specifies the macro to be launched and its parameters:

```
\[MyMacro.PrintLetterTemplate "[Requester.MrMrs]",
 "[Requester.FirstName]", "[Requester.Name]",
"[Requester.Location.Address1]",
"[Requester.Location.Address2]",
"[Requester.Location.ZIP]",
"[Requester.Location.City]", "[ReqNumber]"\]
```

Click Create to confirm the creation of the action.

Step 2: Launching the action

To launch the action:

1. Open the table of purchase requests.
2. Select a purchase request.
3. Launch the action via the **Tools/ Actions** menu item.

When the action is launched:

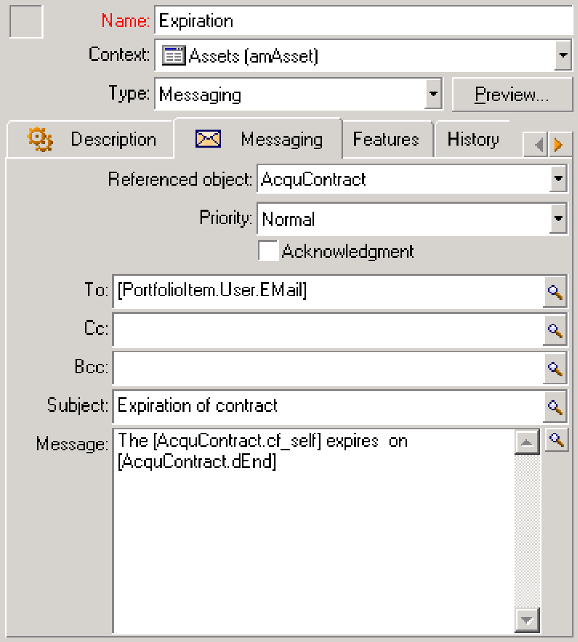
1. Microsoft Word is launched and loads **LetterTemplate.doc**.

2. The details of the person to contact and the request number are inserted in the letter.
3. The letter is printed.

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:

Figure 11.3. Detail of a messaging type action with a referenced object



Name: Expiration
Context: Assets (amAsset)
Type: Messaging

Referenced object: AcquContract
Priority: Normal
□ Acknowledgment
To: [PortfolioItem.User.EMail]
Cc:
Bcc:
Subject: Expiration of contract
Message: The [AcquContract.cf_self] expires on [AcquContract.dEnd]

Example of a Script type action

Creating a **Script** type action basically involves writing the Basic script that modifies the AssetCenter 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 **Tools/ Actions/ Edit** menu item and click the New button in the action detail screen.
2. Assign a name to the action you are going to create, e.g. "Test", and set the **Type** field (SQL name: seActionType) to **Script**. Do not select a context for the action. Click Create.
3. In the **Script** tab, click the button to display the script builder window. The programmable function, called `Success()`, used for these actions does not require an explicit return code. In the following example, we will create a new record in the table of categories based on the information contained in the table below:

Table 11.1. Example of a Script type action

Field label	SQL name of the field	Value of the field
Category	Name	PC
Nature	seNature	Computer
'Connection' tab visible by default	bIsCnxClient	This box is checked

Writing the script

Enter the following example:

```
Dim lrec As Long
Dim lres As Long
  lrec=AmCreateRecord("amCategory")
  lres=AmSetFieldStrValue(lrec, "Name", "Micro")
  lres=AmSetFieldStrValue(lrec, "seNature", 1)
  lres=AmSetFieldStrValue(lrec, "blsCnxClient",
1)
  AmInsertRecord(lrec)
```

Note: This action creates the desired category without any user intervention.

Demonstration of the "Set()" function

Now we will create the same category from a **Script** type action, by specifying the table of categories as the context for the action. In this case, we write the script as follows:

```
Set [Name]="Value"
Set [seNature]=1
Set [blsCnxClient]=1
```

Note: To execute this action, the user must open the table of categories and press . After executing the script, the user must also click 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]**.

For help in entering these variables, click the magnifier 🔍 to the right of the field to be populated.

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**.

Warning: In order for the variables to work, the Context field of the action detail must show a table in AssetCenter and you must select a record in the list of records of the table before executing the action.

Specificities of the Sybase SQL Anywhere database engine

When using Sybase SQL Anywhere as AssetCenter's database engine, it is not possible to write "{d" or "m" at the start of a field in an action.

If you need a field in an action to start with "{d" or "m", we recommend preceding these strings by a space.

Testing an action

To test an action when creating it, use the Preview... found in the top-right corner of the detail of the action to be tested.

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

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 Preview... 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.

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 in order 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.

12 Managing deadlines with AssetCenter Server

CHAPTER

This chapter explains how to manage deadlines (alarms, purchase request approvals, stock reordering, etc.) and automatic triggering of actions (automatic issuing of reminder messages, etc.).

The administrator manages the monitoring of deadlines and the automatic triggering of actions using the AssetCenter Server program, which is independent of AssetCenter.

Overview of AssetCenter Server

AssetCenter includes a system for monitoring deadlines and automatically triggering actions: This program, called AssetCenter Server, functions independently of AssetCenter.

AssetCenter Server automatically monitors all expiration dates in the designated database:

- Alarms (end of term dates of contracts for example).
- Purchase request approvals.

- Stock line reorder levels.
- Rent calculations at the asset and the contract level.
- Lease contract loss value calculations.
- Expense line split operations associated with cost centers.
- Verification of history lines.
- Workflow deadlines.
- Searches for new workflow execution groups.
- Execution of workflow rules.
- Verification of time zones.
- Synchronization of data with AutoCAD.

If justified to do so by the deadlines, AssetCenter Server performs actions, such as issuing reminder messages in the AssetCenter database via the internal messaging system. If necessary, it calculates contract rent, lease contract loss-values, etc.

Warning: When you exit AssetCenter Server, all automatic monitoring functions are suspended if they have not been launched as services.

It is possible to launch AssetCenter Server on several different machines. Therefore, deadlines to monitor and tasks to perform can be shared by different instances of AssetCenter Server. This enables you to improve the performance of AssetCenter Server.

Note: You must make sure that a given task is only performed by one single instance of AssetCenter Server.

You can use the same Login to connect to the database. This login must have administration rights.

Executing AssetCenter Server

Recommendations

AssetCenter Server needs to frequently access the database. This is more than likely to be held on a network.

- If there is a workstation with a high-speed connection to the database, execute AssetCenter Server from this workstation. Monitoring will be carried out for all users.
- If you can only access the database over a low-speed link and if your server operates using Windows, you have the possibility of running AssetCenter Server directly on the server.

In case of modification of the structure of the database

If you modify the structure of the database via AssetCenter Database Administrator or via the **Configure object** command in the shortcut menu, you must disconnect AssetCenter Server from the database and then reconnect.

Launching AssetCenter Server

Manually launching AssetCenter Server

You can launch the AssetCenter Server program from the programs in the **Start** menu or in the AssetCenter program group.

Warning: When you disconnect AssetCenter Server from the database, all monitoring functions and automatic triggering of action are suspended. When you connect to the database under AssetCenter, AssetCenter displays a warning message indicating that AssetCenter Server has not accessed the database in the last hour.

Automatically launching AssetCenter Service as a service

To launch AssetCenter Server as a service:

- Manually launch AssetCenter Server.
- Choose the connection to your database and select the **Use this connection in service mode** option.
- In the Windows configuration panel, select the AssetCenter Server service and set its launch to automatic mode.

From the DOS command prompt

You can automate the launching of AssetCenter Server by using the following command:

```
aamsrv32 -cnx:<connection name>  -login:<login>
-password:<password>
```

Warning: The login is that of an AssetCenter administrator (either "Admin", or a user login with administration rights).

Strings between <> should have no spaces.

Example:

```
aamsrv32 -cnx:BasePeregrine -login:Gerald
-password:Password
```

This command can be placed in a batch file.

Executing AssetCenter Server manually under Windows

Connecting AssetCenter Server to a database

Only an administrator can connect to the database via AssetCenter Server. This can be the "Admin" user or a user with administration rights. Only the administrator can connect to the database via AssetCenter Server.

You need to enter the relevant **Login** and password. Select the **Use this connection in service mode** check box if you want to use this connection as the default connection if AssetCenter Server is running in service mode (as in a service in Windows NT).

To connect AssetCenter Server to the database, use one of the following:

- The dialog box that appears when starting.
- The **File/ Connect** menu item.
- The icon.

Disconnecting AssetCenter Server from a database

To disconnect AssetCenter Server from the database, use one of the following:

- The **File/ Disconnect** menu item.
- The icon.

Using the security-integrated Windows NT

Using the **integrated security** of Windows NT (also called **unified login**) consists of synchronizing the security information from AssetCenter with the information from the Windows NT User Manager (notably, the **login** and **password**).

With this synchronization:

- You can automatically import the list of employees declared in the User Manager into the AssetCenter database.
- The users who connect to an AssetCenter database don't have to populate the **Login** and **Password** fields.

General overview

- AssetCenter Server handles the synchronization between AssetCenter and the User Manager.
- The groups created with the User Manager are used in the attempt to associate an access profile to the AssetCenter users.

- AssetCenter Server uses the **User name** field (User Manager) and **Login** field (AssetCenter) to match the records in the two databases.
- A **Security identifier** feature is created in the AssetCenter database. This feature stores the **SID** of the Windows NT user (**Security identifier**).
- A user is authorized to connect to the AssetCenter database if the following items have the same value:
    - **Security identifier** feature of the AssetCenter user whose login corresponds to the one used to connect to Windows.
    - **SID** associated to the login used to connect to Windows.

Main implementation steps

Here are the main steps you need to take to implement this system. We will elaborate on each one of these steps later on in the guide.

- Create the users and groups in the Windows NT User Manager.

  Objective: Prepare the information to transfer to AssetCenter.
- Create in AssetCenter access profiles that have the same name as the groups in Windows NT.

  Objective: Automatically associate the employees in the AssetCenter database to an access profile having the same name as an NT group.
- Configure AssetCenter Server.

  Objective: Define which NT domains retain and plan the transfers.
- Trigger AssetCenter Server's **Update the table of employees according to the NT users** module.

  Objective: Transfer the information for the first time.
- Complete the description of the employees in the AssetCenter database.

Step 1: Create the users and groups in the Windows NT User Manager

The mandatory fields are:

- For the users: **Full name**.

Tip: When you create users in the AssetCenter database, AssetCenter Server takes the value of the Full name field (from the User Manager) and looks for the first space character starting on the left. Everything it finds to the left of the first space is used to create the First field; everything to the first of the first space creates the Name field. If there is not space, only the Name field will be populated.

- For groups: **Name.**

Step 2: Create access profiles in AssetCenter

During the transfer, if the NT user does not belong to a group whose name is identical to an existent access-profile name in the database, AssetCenter Server creates the user by attaching them to this profile. If this is not the case, the user that is created will not have any rights, and the administrator must manually modify each record.

We suggest then that you first create profiles corresponding to the different NT user groups and associate access rights and restrictions to them. During the creation, the AssetCenter will have the adequate profile right away.

Step 3: Configure AssetCenter Server

AssetCenter Server handles the synchronization of Employees table according to NT users. To carry out this operation:

1. Start AssetCenter Server.
2. Connect to the database on which you want to use the **integrated NT security**, using the **File/ Connect** menu.
3. Select the **Options/ Configure** menu, then click the **Modules** tab in the dialog box. The list of AssetCenter Server modules appears. Each of these modules executes a particular task following a defined periodicity (for more information on how to use AssetCenter Server, consult the Reference manual: **Administration and advanced use of AssetCenter, Managing deadlines with AssetCenter Server** chapter). The module concerned by this function is called **Update**

the table of employees according to the NT users. Make sure that this option is selected.

4. Click the magnifying glass to the right of the **User data item** field. AssetCenter Server displays a complete list of the NT domains and groups on the network.
5. Select from the list the domains containing the users (or user groups) for which you want to activate a **unified login**. Then validate it.
6. Define the execution periodicity of this task. If the information of your NT domain changes frequently, you will want to execute this task regularly. However, you need to keep in mind that the recovery of this information can be very long.
7. Validate your changes by clicking **OK**.

Step 4: Activate the updating process for the first time

1. Select the **Action/ Activate** menu.
2. Check the case corresponding to the **Update the table of employees according to the NT users** task.
3. Click **OK**. AssetCenter begins the update.

Note: If the number of users concerned by this operation exceed the number of Named logins authorized by your license, AssetCenter Server declares the users as having Floating type logins.

Step 5: Complete the description of the employees in the AssetCenter database

Your database now contains one employee per NT user detected on the chosen domain(s). A feature containing the NT security ID (**SID**) is associated to each of these employees. Their login is in the form:

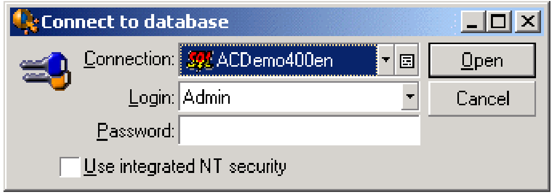
```
[domain]\[user];
```

The users created do not have a valid password.

Note: Following this operation, we advise you to verify that all the records created in the Departments and Employees table correspond to an AssetCenter user. You must notably re-enter the value of the Password field.

AssetCenter connection

Once the previous operation is completed, the NT users can directly access AssetCenter. During their first connection, all they need to do it select the **Use integrated NT security** option, then click **Open**.



Connect to database

Connection: ACDemo400en

Login: Admin

Password:

□ Use integrated NT security

No other authentication information will be asked for the next connections.

If a user wants to connect under another login, however, they can always do so by activating the connection box via the **File/ Connect to database** menu.

Reference information

Supported environments

- Windows domain: Windows NT 4 is supported. Windows Active Directory 2000 is not supported.
- Windows clients: Windows NT, 2000 and XP are supported. Windows ME, 95 and 98 are not supported.

Rules applied by AssetCenter during the creation/modification of users

- The AssetCenter login is created by concatenating the Windows **Domain** name and **User name** in the form of: **<Domain name>\<User name>**
- The **Login** field is used as a reconciliation key for AssetCenter users. This key is unique and thus perfectly reliable.
- When a user is transferred, the user is only created if the login does not yet exist in the database. The **Name**, **First** and **Password** fields, as well as the **Security Identifier** feature and the **Profile** link are populated.

Note: The Password field is populated with a special character that prohibits the user to access the AssetCenter database. You thus must populate this field manually if you want to authorize this user to access the database.

- When a user is imported and their login already exists in the AssetCenter database, the **Name**, **First**, **Password** fields, as well as the **Profile** link, are not modified. Only the value of the **Security identifier** feature is updated, if the NT **SID** is different.
- Remember that there is an index in AssetCenter composed of the **Name**, **First** and **Bar code** fields. This index must be unique. You also need to verify that the **Bar code** field is populated with a different value for each user. This way, you can create different users who might share the very same name. For this, you can use the default value calculation script installed by AssetCenter.

How is an AssetCenter access profile associated with the users?

Note: The association of a profile with a user is only done when the users are created. If the user already exists, the access profile can be neither associated nor modified.

For each user to create in the AssetCenter database:

- For each access profile existing in the AssetCenter database:
    - Look to see if an NT Group having the same name as the access profile belonging to the user.
        - If one exists: The access profile is associated to the user.
        - Otherwise: Examine the following access profile:

Note: This examination of access profiles and NT groups is not ordered.

If you exit AssetCenter Server

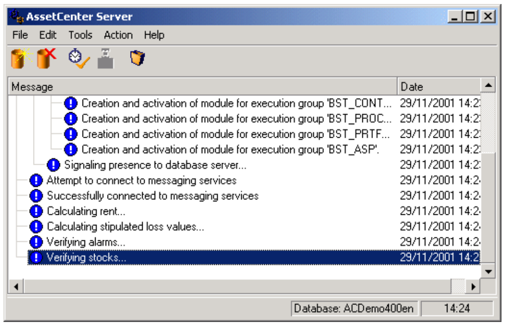
Nevertheless, if AssetCenter Server encounters a connection error after a connection has been opened it, will try to reconnect to the database according to the frequency defined in the general monitoring options. These attempts to reconnect to the database will only be made if the original connection was made without error.

AssetCenter Server and the messaging system

The computers on which AssetCenter Server is installed must have a working messaging system installed. Each login user must be configured correctly in the database in order to send messages via the specified messaging system.

Main screen of AssetCenter Server

Figure 12.1. Main screen of AssetCenter Server



```
Message                                                                     Date
      Creation and activation of module for execution group 'BST_CONT...    29/11/2001 14:2
      Creation and activation of module for execution group 'BST_PROC...    29/11/2001 14:2
      Creation and activation of module for execution group 'BST_PRTF...    29/11/2001 14:2
      Creation and activation of module for execution group 'BST_ASP'.      29/11/2001 14:2
    Signaling presence to database server...                                29/11/2001 14:2
  Attempt to connect to messaging services                                  29/11/2001 14:2
  Successfully connected to messaging services                              29/11/2001 14:2
  Calculating rent...                                                       29/11/2001 14:2
  Calculating stipulated loss values...                                     29/11/2001 14:2
  Verifying alarms...                                                       29/11/2001 14:2
  Verifying stocks...                                                       29/11/2001 14:2

                                               Database: ACDemo400en    14:24
```

The main window displays all events handled by AssetCenter Server.

It enables you to access the following program menus and icons:

Table 12.1. AssetCenter Server program menus and icons

Icon	Menu	Function
	File/ Connect	Connect to a database.
	File/ Disconnect	Disconnect from a database.
	Action/ Activate	Selectively activate deadline monitoring agents.
	Action/ Empty list	Clear messages from main screen.
	Options/ Configure	Configure deadline monitoring options.
	Action/ Connect to messaging	Try to connect to external messaging systems. This button is grayed out if the connection is successful. If the connection fails, this button is available. Click to test the connection.

Whenever the list of events becomes too large, you can use the **Action/ Empty list** command to empty it.

General options of AssetCenter Server

The general monitoring options appear in the **General** tab of the screen displayed by the **Options/ Configure** menu item.

These govern the general operation of AssetCenter Server.

Database reconnection delay

This field is used if AssetCenter Server receives an error message when trying to connect to the database, after the initial connection has been made successfully.

In this case, AssetCenter Server considers that the connection to the database has been lost and stops monitoring deadlines. AssetCenter Server tries to reconnect to the database according to the frequency defined in the **Database** field in the **Reconnection delays** frame.

Monitoring function are resumed when AssetCenter Server manages to reconnect to the database.

The data entry format for this field is "Duration".

Messaging reconnection delay

If a problem occurs with the external messaging system, AssetCenter Server will stop sending external messages.

AssetCenter Server tries to reconnect to the external messaging system according to the frequency defined in this field.

The data entry format for this field is "Duration".

Log file

File

This file stores the messages displayed in the main window of AssetCenter Server.

Max. size

This field enables you to limit the size of the file that logs the messages shown in the main screen of AssetCenter Server.

When this limit is reached, the oldest messages are deleted as and when new messages are recorded.

Time zones

In the **General** tab of the configuration screen, you configure the types of tests 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 AssetCenter Server is installed. The time difference is expressed as [(n * 30minutes) + m] where m is between -15 minutes and + 15 minutes.

In both cases

If the time difference exceeds 5 minutes, AssetCenter Server 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 five minutes (resulting from either a modification to the time of the database server and/or of the machine on which AssetCenter Server, 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 AssetCenter is updated (if the number (n * 30 minutes) does not correspond to the time zone of the server).

To do this, the machine on which AssetCenter Server 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 AssetCenter, is recovered.

Note: Whichever option you select, the tests are performed when AssetCenter Server connects to the database, then regularly according to the frequency defined in the Modules tab of the configuration screen of AssetCenter Server.

Configuring the modules monitored by AssetCenter Server

Introduction

To consult or configure a model monitored by AssetCenter Server, select the **Options/Configure** modules menu item.

The window that appears enables you to display the list of modules and to define for each module:

- If the module is enabled or not.
- Which task the module must perform.
- How frequent each module is triggered.

Tip: You can launch these modules on several machines by executing a AssetCenter Server session on each one of them. This can increase performances. Make sure that a module is only enabled on one machine at a time.

The following sections describe each one of the modules managed by AssetCenter Server. These modules are identified by the description and the name that is listed in AssetCenter Server.

Verification schedules

To define the verification schedules of a module:

1. Select the **Options/ Configure modules** menu.
2. Select the module to configure.
3. Populate the tab to the left of the **Verification schedules** zone.
4. If necessary, create and populate additional sub-tabs by right-clicking the labels of the sub-tabs and selecting the **Add a rule** menu item.

The sub-tabs are used to enable you to define the days and times of monitoring.

Days of monitoring

Table 12.2. Days of monitoring

Value of the Days field	Monitoring is performed
"Daily"	Every day of the year, without exception.
"Day of the week:"	Example Every Monday The weekday selected from an itemized list in the field to the right of the **Days** field.
"Day of the year:"	Example July 20 A day or selection of days. These are defined via the **Day**, **Month** and **Year** check boxes.
"The first" "The second" "The third" "The fourth" "The next to last" "The last"	Example "The first" Friday of each month. "The second" Monday of the month of September. "The next to last" Wednesday of the month of November. "The last" Tuesday of every month throughout 2000. The days of the week defined using the **Day** check box, and for the month(s) and year(s) defined using the **Month** and **Year** check boxes.

Monitoring times

Periodical

You can define two frequencies for the verification of a module that depend on the time of day.

The first **During period** frequency applies to scheduled periods that you can create in two ways:

- Graphically, using the schedule bar. Click and drag on this bar to create a schedule period.

- Directly, by entering an itemized list of values in the field to right of the schedule bar. The following syntax is used:

  ```
  <Start time of period - End time of period>
  ```

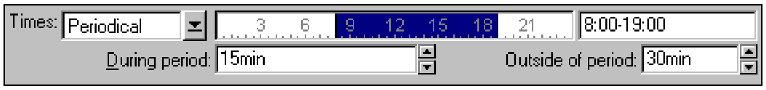
  The time format defined in AssetCenter must be respected.

  To indicate several periods, you just need to separate them with semi-colons ";".

The second **Outside of period** frequency is applied outside of those scheduled periods you have defined.

Example:

Figure 12.2. Time table periods



Times: Periodical

8:00-19:00

During period: 15min

Outside of period: 30min

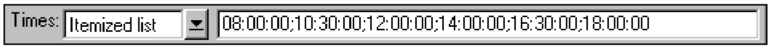
Itemized list

You can enter those times you want the verification to be carried out on in the field to the right.

- The time format defined in AssetCenter must be respected.
- The semi-colon ";" is used to separate them.

  Example:

Figure 12.3. Time table periods



Times: Itemized list

08:00:00;10:30:00;12:00:00;14:00:00;16:30:00;18:00:00

Preview

You can have a preview of:

- A rule defining the monitoring schedules of a module in the **Preview** field of a rule description sub-tab.
- All the rules concerning a module in the **Preview** sub-tab of the **Modules** tab, once the module is selected.

Verify history lines module (History)

Sometimes when a record is destroyed in the database, the corresponding history lines are not destroyed. AssetCenter Server verifies if there are any such history lines; if it finds any it destroys them.

Verify stocks module (Stock)

AssetCenter Server monitors stock reorder levels.

For each stock, AssetCenter Server refers to the stock rules defined in the **Manage** tab of the stock detail.

For each stock rule concerning a model:

- AssetCenter Server calculates the quantity of items actually available from the **Assignments** field in the detail of a portfolio item.
- When the quantity falls below the value specified in the **Reorder level** (SQL name: lReordLevel) field of the stock rule detail, AssetCenter Server automatically creates a purchase request.
    - The parameters of the purchase request can be found in the **Auto-request** and **Management** tabs of the detail of the stock.
    - The purchase request specifies the quantity to be reordered (**To order** field (SQL name: lQtyToOrder) in the detail of the stock rule).
- For as long as the request is not fully received, AssetCenter Server does not verify the stock rule that it has generated. Therefore, no new request is sent.
- As soon as delivery of the request is taken in full, AssetCenter Server:
    - Readjusts the stock levels.
    - Erases the contents of the **Request line** field (SQL name: ReqLine) in the stock rule detail.
    - Reactivates the stock rule.

Verify alarms module (Alarms)

List of alarms monitored

At the asset level

Several key dates are monitored:

- The end-of-reservation date of an asset: This is shown in the **Reserv. end date** field (SQL name: dtEnd) in the **Portfolio/Reservations** tab of the asset detail.
- The warranty expiration date of an asset: Asset detail, **Maint.** tab, **Expiration** field (SQL name: dWarrEnd).
- End-of-term date for lease, rental, loan of an asset: This alarm can only be defined if the acquisition method of the asset (Asset detail, **Acquis.** tab, **Acq. method** field (SQL name: seAcquMethod)) is set to **Lease**, **Rental** or **Loan**. In this case the **Price and conditions** sub-tab of the **Acquis.** tab shows an **End date** field (SQL name: dEndAcqu).
- End-of-rent dates of an asset: Alarms can be attached to end of validity dates (**Acquis.** tab, rent descriptions sub-tabs, **Schedule** frame).

At the consumable level

AssetCenter Server monitors the end-of-reservation date for consumables: This is shown in the **Reserv. end date** field (SQL name: dReservEnd) in the reservation detail of a consumable. To access the reservation detail of a consumable:

1. Launch AssetCenter.
2. Select **Procurement/ Purchase requests**.
3. Select the purchase request reserving the consumable.
4. Display the composition of this purchase request.
5. Display the request line corresponding to the consumable.
6. Display the **Reservations** tab of the request line. This tab shows the list of reservations for consumables.
7. Display the detail of the reservation.

The monitored field is **Date fin** (Nom SQL : dtEnd).

At the project level

AssetCenter Server monitors the end dates of project: Project detail, **General** tab, **End** field (SQL name: dEnd).

At the contract level

Several key dates are monitored:

- The end-of-term date: Contract detail, **General** tab, **End** field (SQL name: dEnd).
- If the contract **Type** (SQL name: seType) is **Lease schedule** or **Master lease**: Alarms can be attached to the notification dates for possible end of lease. These dates are shown to the right of the notification field in the sub-tabs describing the possible end of term options: **Renewal**, **Purchase**, **Return**.
- If the contract **Type** (SQL name: seType) is **Lease schedule**: Alarms can be attached to the end dates of validity for rent items as shown in the individual rent-description sub-tabs of the **Rent** tabs.

At the purchase request level

If the acquisition method of the purchase request (Purchase request detail, **Financing** tab, **Acq. method** field (SQL name: seAcquMethod)) is set to **Lease**, **Rental** or **Loan**, it is possible to define an alarm associated with the rental, lease or loan end dates (**Acq. method** field in **Financing** tab of purchase request detail).

The same is true for estimates and orders.

What happens in two-level alarms when the first level action has been triggered?

In the case of alarms with 2 levels, the triggering of the second level alarm depends on the action carried out at the first level.

- If the first-level alarm triggers an action other than the sending of a message via AssetCenter's internal messaging system (such as sending a message via a third-party messaging system), the second-level alarm will always be triggered at the defined moment.

- If the first level-alarm sends a message to a group of AssetCenter users via the internal messaging system, the action defined at the second level will not be triggered if one or more of the recipients has read the message.

Calculate rents module (Rent)

AssetCenter Server monitors periodic rent payments for contracts and assets. It calculates and/recalculates the amounts involved.

The **Calculate rents** module defines:

- Certain parameters concerning the generation of costs for contracts and asset-level rent payments.
- The frequency of updates.

Overview

AssetCenter Server verifies at regular intervals whether it needs to generate expense lines. If this is so, it generates them.

After checking and generating the expense lines relative to a periodic rent, AssetCenter Server stores the date of the last expense line (past or present) in the **Recalculation effective from** field (SQL name: dRecalcul).

- If the contract-level rent is distributed to the assets, AssetCenter Server modifies the **Recalculation effective from** field that is found in the rent sub-tabs of the **Acquis.** tab of the assets detail.
- If the contract-level rent is not distributed to asset level, AssetCenter Server modifies the **Recalculation effective from** field, which is found in the rent sub-tabs of the **Rents** tab of the contract detail.

AssetCenter Server does not recalculate every single expense line each time.

- Projected expense lines associated with a periodic rent are always recalculated.
- The **Recalculation effective from** field, proper to each rent, sets the date from which past and present expense lines associated with a periodic rent are recalculated.

The lessee may directly modify the recalculation date of the non-projected expense lines by directly modifying the **Recalculation effective from** field. This flexibility enables you to recalculate erroneous expense lines in case of a change in tax rates, for example.

Parameters

The **User data item** field is used to set the rent generation parameters. The syntax of this field is as follows:

```
<Duration>d
```

This duration set the number of days for which the calculation is made. For example, if you want to calculate rent over a period of 90 days, enter the following value:

```
90d
```

Note: The maximum number of rent calculations made per transaction is specified by the UserData entry in the Amsrv.cfg configuration file.

Projected rent

The **User data item** field enables you to specify the number of days for which you calculate project rent payments.

AssetCenter Server generates the projected expense lines for the specified period. In order to not generate any, you just need to set this field to **0**.

Example

Let's consider the following configuration:

- The contract is effective from July 1, 2001 through July 1, 2004.
- The rent is payable monthly on the first day of the month.
- AssetCenter Server verifies rent payments every 2 months and generates projected rent payments for the next 12 months.

On July 1, 2002, AssetCenter Server is launched for the first time: it generates:

- Past rents from July 1, 2001 through July 1, 2002.
- The present rent on July 1, 2002.
- The projected rents from August 1, 2002 through July 1, 2003.

Following these calculations, the Recalculation effective from field indicates the date of the last non-projected expense line, i.e. July 1, 2002.

AssetCenter Server runs in the background: 2 months later on September 1, 2002, it generates:

- The projected rents from October 1, 2002 through September 1, 2003.
- Past or present rents for which the payment date is later than that contained in the Recalculation effective from field, i.e. the rents from August 1, 2002 through October 1, 2002.

Calculate stipulated loss values module (LostVal)

AssetCenter Server recalculates, at regular intervals, the loss values for lease contracts whose calculation method is set to **Calculate for all periods** (**Calculation** field (SQL name: seLossValCalcMode) in **Leasing** tab of the lease contract detail). In this way, loss values pertaining to any loss value rules that have been modified since the last time AssetCenter Server accessed the database are updated.

Split expense lines in cost centers module (CostCenter)

AssetCenter Server handles split operations for expense lines.

Overview

AssetCenter Server searches the expense lines to be split: These are the expense lines whose **Split operation status** field (SQL name: seSplitStatus) is set to **Not split**.

By default, all expense lines are to be split, regardless of their status (**Status** field (SQL name: seStatus) of an expense line).

AssetCenter Server splits the designated expense lines. When an expense line is split:

- A debit expense line, equivalent to the split expense line is created in the parent cost center.
- Expense lines are created in the target cost centers, according to the split percentage values. By default, these are **Not split**.

Specific example: Managing the removal of a cost center

When you decide to delete a cost center, and the cost center contains expense lines, AssetCenter will not allow you to perform the operation unless the **Authorize extended deletions** option in the **General** tab in the **Tools/ Options** menu item is checked.

In this case, AssetCenter gives you three possibilities:

- Delete all the linked records.
- Detach the linked records.
- Attach the linked records to another record.

What happens next depends on the option you choose:

Delete all linked records

When a cost center is deleted, AssetCenter deletes:

- The expense lines of the deleted cost center.
- The expense lines resulting from split operations on the deleted cost center.

An AssetCenter agent modifies the Split operation **Split operation status** field (SQL name: seSplitStatus) in order for it to show "Not split" at the level of the expense lines highest up in the split operation, and which, when split, generated the expense lines belonging to the deleted cost center (after any intermediate split operations).

When AssetCenter Server finds these expense lines, which are not split but have generated split expense lines, it deletes all the expense lines resulting from their split operations. In doing this, AssetCenter Server deletes the expense lines that, when split, generated the expense lines belonging to the deleted cost center.

Then AssetCenter Server performs the split operations on those expense lines, which have not yet been split. It thus recalculates, using new parameters, all the expense lines that, when split, generated the expense lines of the deleted cost center.

Detach all linked records

In this case:

- The expense lines of the deleted cost center are no longer associated with a cost center.
- The expense lines, which when split generated the expense lines for the deleted cost center are split again.
- The expense lines, resulting from split operations on the deleted cost center, are not modified.

Attach linked records to another record

In this case, you select another cost center X, which takes the place of the deleted cost center:

- The expense lines of the deleted cost center are attached to cost center X.
- The expense lines, which when split generated the expense lines for the deleted cost center, are split again; cost center X is considered as the new target cost center.
- The expense lines resulting from split operations on the deleted cost center are deleted and the expense lines of cost center X are split.

Signal presence to database server module (UpdateToken)

AssetCenter Server regularly sends a signal to the database server in order to indicate that it is functioning.

If the database server does not receive a signal from AssetCenter Server for over one hour, a message is displayed when a user connects to the database under AssetCenter.

This message indicates that AssetCenter Server has not been launched on this database for over one hour and that without this process, monitoring functions will be interrupted.

If the database server goes without receiving a signal from AssetCenter Server for over a week, it is no longer possible to connect to the database.

Search for new workflow execution groups module (WorkflowFinder)

AssetCenter Server monitors the creation of new workflow execution groups.

As soon as AssetCenter Server detects a new workflow execution group **G**, it creates a new monitoring module **Execution of workflow rules for execution group G**.

This mechanism has the following advantages:

- It enables you to define verification timetables for each workflow execution group.
- Different workflow execution groups can be monitored by different instances of AssetCenter Server.

Execute workflow rules for execution group 'G' modules

Once a workflow execution group **G** is detected, AssetCenter Server executes the appropriate workflow rules.

Monitoring of workflow execution groups

AssetCenter Server monitors the deadlines specific to workflow instances associated with the execution group.

Deadlines to be monitored by AssetCenter Server as soon as the activity is triggered are defined in the **Alarms** tab of the detail of the workflow activity.

These deadlines are defined by the time limits defined for the set tasks to be carried out.

Note: In the case of deadlines specific to Workflow, business calendars specified in the Time limit tab in the activity detail are taken into account. When calculating deadlines, these time limits are converted to business hours.

Processing of Periodical type events

According to the frequency defined in the **Parameters** tab in the detail of a **Periodical** type event, AssetCenter Server triggers the event if the activation conditions are met.

Then, the role of AssetCenter Server depends on the mode of processing of the event as indicated in the **General** tab of the event detail:

- **Log event and process by server**: As soon as the event occurs, AssetCenter Server saves it to the table with SQL name "amWfOccurEvent".

  Then, AssetCenter Server activates the transition according to the frequency of verification as defined in the configuration screen of AssetCenter Server.
- **Log event and process immediately**: As soon as the event occurs, AssetCenter Server saves it to the table with SQL name "amWfOccurEvent", and activates the transition.
- **Process event immediately without logging**: As soon as the event occurs, AssetCenter Server activates the transition.

Activation of transitions

AssetCenter Server activates the transitions for events according to the frequency defined in the configuration screen. The following events are concerned:

- **System** events.

- **Database** and **Periodical** type events whose processing mode is set to **Log event and process by server**.

Execution of tasks

AssetCenter Server executes tasks resulting from **Automatic action** or **Test / script** type activities, except in the possible case of tasks resulting from activities whose **Execute actions immediately** (SQL name: bExecImmediately) box is selected.

The frequency with which AssetCenter Server verifies and performs the tasks it has to carry out is indicated in the configuration screen of AssetCenter Server.

In the case of a task originating from an **Automatic action** or **Test / script** type activity whose **Execute actions immediately** box (SQL name: bExecImmediately) is checked:

- This task is executed by AssetCenter Server if it is AssetCenter Server that activates the transition creating the task. In this case, AssetCenter Server performs the task as soon as the transition it creates is activated.
- Otherwise, the AssetCenter client machine executes the task.

Add the computers listed in the NT domain to the database module (AddCpu)

AssetCenter Server enables you to program the recovery of those computers declared in the NT domain.

The domain to analyze is specified at the Connect-It **addcpu.scn** scenario.

Before activating the **Add the computers listed in the NT domain to the database** module, you need to verify that the following parameters are correct:

- Parameters of the AssetCenter Server module **Add the computers listed in the NT domain to the database.**
- Parameters of the Connect-IT **addcpu.scn** scenario, which is located in the **scenario\ntsec\ntac40** sub-folder of the Connect-It installation folder.

Parameters of the module Add the computers listed in the NT domain to the database (User data item field).

Here is the parameter line by default:

```
"$connectit_exedir$/conitsvc.exe" -once
'$connectit_exedir$/../scenario/ntsec/ntac40/addcpu.scn'
-d:AssetCenter.SERVER=$cnx$
-d:AssetCenter.LOGIN=$login$
-d:AssetCenter.TEXTPASSWORD=$pwd$
```

Information about certain parameters in this line:

- `$connectit_exedir$` stores the path of the **conitsvc.exe** program in the Windows Registry.

  You have no need to modify this parameter.
- **-once** indicates that **conitsvc.exe** must be executed one time (in other words, using Connect-It's **Once** scheduler.).

  Do not modify this parameter; you define the programming at the level of the AssetCenter application.
- `$connectit_exedir$/../scenario/ntsec/ntac40/addcpu.scn` is the path to the Connect-It scenario to use..

  Modify this parameter if you want to use another Connect-It scenario.
- `-d:AssetCenter.SERVER=$cnx$`
  `-d:AssetCenter.LOGIN=$login$`
  `-d:AssetCenter.TEXTPASSWORD=$pwd$` stores the AssetCenter connection name opened by AssetCenter Server, as well as the login and password used for the connection.

  These parameters overwrite the values defined at the level of the AssetCenter connector in the **addcpu.scn** scenario.

  You have no need to modify these parameters.

Parameters of the addcpu.scn Connect-It scenario

To modify the **addcpu.scn** scenario:

1. Execute the Connect-It scenario editor

2. Open the **addcpu.scn** scenario, which is located in the **scenario\ntsec\ntac40** sub-folder of the Connect-It installation folder.
3. Select the **NT Security** connector in the **Scenario diagram** window by clicking on the title bar of the **NT security** box; not inside of the box itself.
4. Select the **Tools/ Configure** menu.
5. Click **Next**
6. Populate the **Domain(s)** field with the name of the domain from which you want to import the computers.

Warning: The computers that can be recovered are:

- Those with the same domain as the one to which AssetCenter Server is connected.
- Those domains which are **trusted** by the one to which the AssetCenter Server is connected.

Tip: To find out whether or not a computer has been recovered:

1. Execute the Windows explorer on the AssetCenter Server machine.
2. Display the network neighborhood.
3. The computers that you see are those from which AssetCenter Server could recover information.

To find out which information is populated in the AssetCenter database, examine in detail the **addcpu.scn** scenario.

The module populates the **Next scan** field (dtNextScan) in the **Computers** table (amComputer) with the date of the module's execution at 00:00.

To learn how the **NT Security** connector works, refer to the Connect-It **User's guide**, chapter **Application connectors**, section **NT Security connector**.

To learn how the **AssetCenter** connector works, refer to the Connect-It **User's guide**, chapter **Peregrine Systems connectors**, section **Asset Management connector**.

Add NT users to the database module (AddUser)

AssetCenter Server enables you to program the recovery of the users declared on the NT domain.

This is essentially used to populate the **Departments and employees** table with the information useful for connecting to an AssetCenter database that uses integrated NT security.

The domain to analyze is specified at the Connect-It **adduser.scn** scenario.

Before activating the **Add NT users to the database** module, you need to verify that the following parameters are correct:

- Parameters of the AssetCenter Server module **Add NT users to the database**.
- Parameters of the Connect-It **adduser.scn** scenario, which is located in the **scenario\ntsec\ntac40** sub-folder of the Connect-It installation folder.

Parameters of the module Add NT users to the database (User data item field).

Here is the parameter line by default:

```
"$connectit_exedir$/conitsvc.exe" -once
'$connectit_exedir$/../scenario/ntsec/ntac40/adduser.scn'
-d:AssetCenter.SERVER=$cnx$
-d:AssetCenter.LOGIN=$login$
-d:AssetCenter.TEXTPASSWORD=$pwd$
```

Information about certain parameters in this line:

- `$connectit_exedir$` stores the path of the **conitsvc.exe** program in the Windows Registry.

  You have no need to modify this parameter.
- **-once** indicates that **conitsvc.exe** must be executed one time (in other words, using Connect-It's **Once** scheduler.).

  Do not modify this parameter; you define the programming at the level of the AssetCenter application.
- `$connectit_exedir$/../scenario/ntsec/ntac40/adduser.scn` is the path to the Connect-It scenario to use..

  Modify this parameter if you want to use another Connect-It scenario.
- `-d:AssetCenter.SERVER=$cnx$`
  `-d:AssetCenter.LOGIN=$login$`
  `-d:AssetCenter.TEXTPASSWORD=$pwd$` stores the AssetCenter connection name opened by AssetCenter Server, as well as the login and password used for the connection.

  These parameters overwrite the values defined at the level of the AssetCenter connector in the **adduser.scn** scenario.

  You have no need to modify these parameters.

Parameters of the addcpu.scn Connect-It scenario

To modify the **adduser.scn** scenario:

1. Execute the Connect-It scenario editor
2. Open the **adduser.scn** scenario, which is located in the **scenario\ntsec\ntac40** sub-folder of the Connect-It installation folder.
3. Select the **NT Security** connector in the **Scenario diagram** window by clicking on the title bar of the **NT security** box; not inside of the box itself.
4. Select the **Tools/ Configure** menu.
5. Click **Next**
6. Populate the **Domain(s)** field with the name of the domain from which you want to import the computers.

Tip: If you want to explore several domains, we recommend creating a Connect-It scenario and AssetCenter Server module per domain.

Warning: The users that can be recovered are:

- Those on the same domain as the one to which the AssetCenter Server user is connected.
- Those domains that are **trusted** by the one to which the AssetCenter Server user is connected.

Tip: To find out if a machine can be recovered:

1. Execute the Windows explorer on the AssetCenter Server computer.
2. Make a folder into a shared folder.
3. Define the permissions for this shared folder.
4. Add a user to the permissions.
5. The users that you see are those from which AssetCenter Server could recover information.

To find out which information is populated in the AssetCenter database, examine in detail the **adduser.scn** scenario.

To learn how the **NT Security** connector works, refer to the Connect-It **User's guide**, chapter **Application connectors**, section **NT Security connector**.

To learn how the **AssetCenter** connector works, refer to the Connect-It **User's guide**, chapter **Peregrine Systems connectors**, section **Asset Management connector**.

Create assets, consumptions, etc. corresponding to items received module (Delivery)

Prerequisites

This module cannot be executed unless you have already done the following:

- Execute AssetCenter
- Select the **Tools/ Database options** menu item.
- Select the **Procurement/Have the application server (aamsrv) create the items received** option.
- Set this option to **Yes**

Task performed by the module

This module is used to process the records from the **Items received** table in order to create received items (assets, consumptions, etc.) in their respective tables.

Utility of this mode

Assigning this task to AssetCenter Server rather than the AssetCenter application can increase the performances of those users receiving orders.

Frequency of execution

We recommend that you execute this module several times by day if you want the users to be able to quickly access the items received in their respective tables.

Send the scanner to the computers module (SendScan)

This module enables you to download and execute the InfraTools Desktop Discovery **scan32.exe** inventory program on a selection of workstations.

The **scan32.exe** program enables you to create an **.fsf** file that stores the results of the machine scan.

The list of machines on which the **scan32.exe** file is copied is defined by taking all the records from the **Computers** table (amComputer) whose **Next scan** field (dtNextScan) is not empty and is inferior to the module's execution date.

Note: This module is based on the assumption that the list of machines has already been created using the Add the computers listed in the NT domain to the database module.

Before activating the **Send the scanner to the computers** module, you need to verify that the following parameters are correct.

Note: The default parameters of the Connect-It sendscan.scn scenario, located in the scenario\autoscan\ac40 sub-folder of the Connect-It installation folder, don't need to be modified in order for the Send the scan to the computers module to work.

Parameters of the module Send the scanner to the computers (User data item field).

Here is the parameter line by default:

```
"$connectit_exedir$/conitsvc.exe" -once
'$connectit_exedir$/../scenario/autoscan/ac40/sendscan.scn'
-d:AssetCenter.SERVER=$cnx$
-d:AssetCenter.LOGIN=$login$
-d:AssetCenter.TEXTPASSWORD=$pwd$
```

Information about certain parameters in this line:

- `$connectit_exedir$` stores the path of the **conitsvc.exe** program in the Windows Registry.

  You have no need to modify this parameter.

- **-once** indicates that **conitsvc.exe** must be executed one time (in other words, using Connect-It's **Once** scheduler.).

  Do not modify this parameter; you define the programming at the level of the AssetCenter application.
- `$connectit_exedir$/../scenario/autoscan/ac40/sendscan.scn` is the path to the Connect-It scenario to use.

  Modify this parameter if you want to use another Connect-It scenario.
- `-d:AssetCenter.SERVER=$cnx$`
  `-d:AssetCenter.LOGIN=$login$`
  `-d:AssetCenter.TEXTPASSWORD=$pwd$` stores the AssetCenter connection name opened by AssetCenter Server, as well as the login and password used for the connection.

  These parameters overwrite the values defined at the level of the AssetCenter connector in the **sendscan.scn** scenario.

  You have no need to modify these parameters.

Parameters of the sendscan.scn Connect-It scenario

To modify the **sendscan.scn** scenario (this should not be necessary):

1. Execute the Connect-It scenario editor
2. Open the **sendscan.scn** scenario, which is located in the **scenario\autoscan\ac40** sub-folder of the Connect-It installation folder.
3. Select the **Send scan** connector in the **Scenario diagram** window by clicking the title bar of the **Send scan** box, but not inside the box itself.
4. Select the **Tools/ Edit mapping** menu.
5. Select the **amComputerSrc-CommandDst/ Command (CommandDst)** line.
6. Click the **magnifier**.
7. Select the **Arg** element.
8. Here is the default mapping script:

```
Dim strArg As String
strArg = "-cmd:scan -certiffile:iftscan.cfg
```

```
-certifpwd:icwpwd"
strArg = strArg & " -ipaddr:" & [TcpIpHostName]
strArg = strArg & " -ipport:738"
strArg = strArg & " -pwd:password"
strArg = strArg & "
-locfile:scanners/scanw32.exe"
strArg = strArg & " -remfile:c:/scanw32.exe"
strArg = strArg & " -remparam:-oC:\" &
[WorkGroup] & "_" & [Name] & ".fsf"
```

Here are some indications on the script parameters:

- `-certiffile:iftscan.cfg -certifpwd:icwpwd` These files are installed in the **bin32\scanners** subfolder of the Connect-It installation folder by the **Automatic inventory** package, supplied on the AssetCenter CD-ROM. They go with the **scanw32.exe** inventory program. They grant you the rights necessasry to scan the remote computer.
- `-ipport:738` and `-pwd:password`: These parameters indicate the port and the password of the InfraTools Remote Control agents on the remote computers. Check that the agents on the remote computers are using this port and password or modify these parameters accordingly. If you want to use a different port and password on certain remote computers, modify the mapping script. The script is written in Basic.
- `-locfile:`: This parameter defines the full path of the **scanw32.exe** file, installed by the **Automatic inventory** package, supplied on the AssetCenter CD-ROM.
- `-locfile:`: This parameter indicates the full path to where the **scanw32.exe** file should be copied on the remote computer.
- `-remparam:`: This parameter defines the full path of the**.fsf** file, produced when **scanw32.exe** is executed on the remote computer. By default, this file is created at the root of **c:\**, and the name of this file is made up of the domain name followed by the name of the remote computer.

Warning: This mapping script must perfectly match the mapping script of the getfsf.scn scenario.

To find out which information is populated in the AssetCenter database, examine in detail the **sendscan.scn** scenario.

The module populates the **Next scan** field (dtNextScan) in the **Computers** table (amComputer) with:

- If the scan was performed successfully: the date of the module's execution + 7 days.
- If the scan was performed unsuccessfully: the date of the module's execution + 1 day.

Tip: A history of scans performed successfully or unsuccessfully is kept in the Scans field (ScanHistory) in the Computers table (amComputer).

To learn how the **Command line** connector works, refer to the Connect-It **User's guide**, chapter **Protocol type connectors**, section **Command line connector**.

To learn how the **AssetCenter** connector works, refer to the Connect-It **User's guide**, chapter **Peregrine Systems connectors**, section **Asset Management connector**.

Troubleshooting

If you have any difficulties in executing this module, make sure of the following:

- An InfraTools Remote Control Agent is installed on the remote computer.
- The port and password of the InfraTools Remote Control Agent are identical to those in the **sendscan.scn** scenario mapping.

- The name of the remote computer is the same at the level of the InfraTools Remot Control Manager, and in the **Name** field (Name) in the **Computers** table (amComputer).
- The TCP/IP layers are correctly installed on the remote computer.

To scan computer that are not running 32-bit Windows

The **Send the scanner to the computers** module only enables you to scan 32-bit Windows computers. To scan computers running other platforms, you must manually use one of the scanners installed by the **Automatic inventory** package, provided on the AssetCenter CD-ROM.

These scanners are created in the **bin32\scanners** subfolder of the Connect-It installation folder. The names of these scanners enable you to identify the platform for which they are intended.

To find out what is inventoried by the scanners

The **Automatic inventory** package, installed from the AssetCenter CD-ROM installs the **settings.txt** file in the **bin32\scanners** subfolder of the Connect-It installation folder.

This file contains a description of what the scanners, located in the same subfolder, inventory during their execution.

Get the results of the scanners module (GetFsf)

AssetCenter Server enables you to program the recovery of the **.fsf** files produced by InfraTools Desktop Discovery (these files store the results of the machine scan).

The list of machines on which the **.fsf** files are recovered is defined by taking all the records from the **Computers** table (amComputer) whose **Last hardware scan** field (dtHardScan) is inferior to the **Last scan** field (dtLastScan).

Note: This module is based on the assumption that the machine scan has already been performed using the Send the scanner to the computers module.

Before activating the **Get the results of the scanners** module, you need to verify that the following parameters are correct:

- Parameters of the AssetCenter Server module **Get the results of the scanners**.
- Parameters of the Connect-It **getfsf.scn** scenario, which is located in the **scenario\autoscan\ac40** sub-folder of the Connect-It installation folder.

Parameters of the module Get the results of the scanners (User data item field).

Here is the parameter line by default:

```
"$connectit_exedir$/conitsvc.exe" -once
'$connectit_exedir$/../scenario/autoscan/ac40/getfsf.scn'
-d:AssetCenter.SERVER=$cnx$
-d:AssetCenter.LOGIN=$login$
-d:AssetCenter.TEXTPASSWORD=$pwd$
```

Information about certain parameters in this line:

- `$connectit_exedir$` stores the path of the **conitsvc.exe** program in the Windows Registry.

  You have no need to modify this parameter.
- **-once** indicates that **conitsvc.exe** must be executed one time (in other words, using Connect-It's **Once** scheduler.).

  Do not modify this parameter; you define the programming at the level of the AssetCenter application.
- `$connectit_exedir$/../scenario/autoscan/ac40/getfsf.scn` is the path to the Connect-It scenario to use..

  Modify this parameter if you want to use another Connect-It scenario.
- `-d:AssetCenter.SERVER=$cnx$`
  `-d:AssetCenter.LOGIN=$login$`
  `-d:AssetCenter.TEXTPASSWORD=$pwd$` stores the AssetCenter connection name opened by AssetCenter Server, as well as the login and password used for the connection.

These parameters overwrite the values defined at the level of the AssetCenter connector in the **getfsf.scn** scenario.

You have no need to modify these parameters.

Parameters of the getfsf.scn Connect-It scenario

To modify the **getfsf.scn** scenario:

1. Execute the Connect-It scenario editor
2. Open the **getfsf.scn** scenario, which is located in the **scenario\autoscan\ac40** sub-folder of the Connect-It installation folder.
3. Select the **Fet FSF** connector in the **Scenario diagram** window by clicking on the title bar of the **Get FSF** box; not inside of the box itself.
4. Select the **Tools/ Edit mapping** menu.
5. Select the **amComputerSrc-CommandDst/ Command (CommandDst)** line.
6. Click the **magnifier**.
7. Select the **Arg** element.
8. Here is the default mapping script:

```
Dim strArg As String
strArg = "-cmd:getfile -certiffile:iftscan.cfg
-certifpwd:icwpwd"
strArg = strArg & " -ipaddr:" & [TcpIpHostName]
strArg = strArg & " -ipport:738"
strArg = strArg & " -pwd:password"
strArg = strArg & " -remfile:c:/" & [WorkGroup]
 & "_" & [Name] & ".fsf"
strArg = strArg & " -locfile:../fsf/" &
[WorkGroup] & "_" & [Name] & ".fsf"
```

Here are some indications on the script parameters:

- `-certiffile:iftscan.cfg -certifpwd:icwpwd` These files are installed in the **bin32/scanners** subfolder of the Connect-It installation folder by the **Automatic inventory** package, supplied on the AssetCenter CD-ROM. They go with the **scanw32.exe** inventory program. They grant you the rights necessasry to scan the remote computer.

- `-ipport:738` and `-pwd:password`: These parameters indicate the port and the password of the InfraTools Remote Control agents on the remote computers. Check that the agents on the remote computers are using this port and password or modify these parameters accordingly. If you want to use a different port and password on certain remote computers, modify the mapping script. The script is written in Basic.
- `-remfile`: This parameter indicates the full path of the **.fsf** files to recover from the remote computers. By default, you'll notice that these files are created at the root **c:\**, and that the names of these files are made up of the domain name, followed by the names of the remote computers.
- `-locfile:`: This parameter defines the full path of the **.fsf** files, copied onto the computer running Connect-It. By default, these files are created in the **\fsf** subfolder of the Connect-It installation folder, and that the names of these files are composed of the domain name followed by the names of the remote computers.

Warning: This mapping script must perfectly match the mapping script of the sendscan.scn scenario.

To find out which information is populated in the AssetCenter database, examine in detail the **getfsf.scn** scenario.

To learn how the **Command line** connector works, refer to the Connect-It **User's guide**, chapter **Protocol type connectors**, section **Command line connector**.

To learn how the **AssetCenter** connector works, refer to the Connect-It **User's guide**, chapter **Peregrine Systems connectors**, section **Asset Management connector**.

Troubleshooting

If you have any difficulties in executing this module, make sure of the following:

- An InfraTools Remote Control Agent is installed on the remote computer.
- The port and password of the InfraTools Remote Control Agent are identical to those in the **sendscan.scn** scenario mapping.
- The name of the remote computer is the same at the level of the InfraTools Remot Control Manager, and in the **Name** field (Name) in the **Computers** table (amComputer).
- The TCP/IP layers are correctly installed on the remote computer.

Update the database with the results of the scanners module (IddAc)

AssetCenter Server enables you to program the recovery of the **.fsf** files produced by InfraTools Desktop Discovery (these files store the results of the machine scan).

The folder containing the **.fsf** files is specified at the level of the Connect-It **iddac.scn** scenario.

Note: This module is based on the assumption that the machine scan has already been performed.

Before activating the **Update the database with the results of the scanners** module, you need to verify that the following parameters are correct:

- Parameters of the AssetCenter Server module **Update the database with the results of the scanners**.
- Parameters of the Connect-It **iddac.scn** scenario, which is located in the **scenario\idd\iddac40** sub-folder of the Connect-It installation folder.

Parameters of the module Update the database with the results of the scanners (User data item field).

Here is the parameter line by default:

```
"$connectit_exedir$/conitsvc.exe" -once
'$connectit_exedir$/../scenario/idd/iddac40/iddac.scn'
-d:AssetCenter.SERVER=$cnx$
-d:AssetCenter.LOGIN=$login$
-d:AssetCenter.TEXTPASSWORD=$pwd$
```

Information about certain parameters in this line:

- `$connectit_exedir$` stores the path of the **conitsvc.exe** program in the Windows Registry.

  You have no need to modify this parameter.
- **-once** indicates that **conitsvc.exe** must be executed one time (in other words, using Connect-It's **Once** scheduler.).

  Do not modify this parameter; you define the programming at the level of the AssetCenter application.
- `$connectit_exedir$/../scenario/idd/iddac40/iddac.scn` is the path to the Connect-It scenario to use..

  Modify this parameter if you want to use another Connect-It scenario.
- `-d:AssetCenter.SERVER=$cnx$`
  `-d:AssetCenter.LOGIN=$login$`
  `-d:AssetCenter.TEXTPASSWORD=$pwd$` stores the AssetCenter connection name opened by AssetCenter Server, as well as the login and password used for the connection.

  These parameters overwrite the values defined at the level of the AssetCenter connector in the **iddac.scn** scenario.

  You have no need to modify these parameters.

Parameters of the iddac.scn Connect-It scenario

To modify the **iddac.scn** scenario:

1. Execute the Connect-It scenario editor
2. Open the **iddac.scn** scenario, which is located in the **scenario\idd\iddac40** sub-folder of the Connect-It installation folder.

3 Select the **Desktop Discovery** connector in the **Scenario diagram** window by clicking on the title bar of the **Desktop Discovery** box; not inside of the box itself.

4 Select the **Tools/ Configure** menu.

5 Click **Next**

6 Populate the following fields:

- **FSF files folder** field: the folder path that contains the **.fsf** files to get (machine scan files).

  This folder must be the same as the one specified in the **getfsf.scn** scenario **mapping**.
- **SAI files folder**: the folder path that contains the **.sai** files (correspondence files between the scanned software files and the corresponding software).

  This folder must be the same as the one specified in the **getfsf.scn** scenario **mapping**.
- **User entry file**: the path of the **.cdt** file. (This Connect-It file lists of fields present in the InfraTools Desktop Discovery scan formula in order that they be present in the Connect-It mapping.)

To find out which information is populated in the AssetCenter database, examine in detail the **iddac.scn** scenario.

The module populates the **Last hardware scan** (dtHardScan), **Last network scan** (dtNetworkScan) and **Last software scan** (dtSoftScan) fields in the **Computers** table (amComputer) with the date of the scan.

To learn how the **Desktop Discovery** connector works, refer to the Connect-It **User's guide**, chapter **Peregrine Systems connectors**, section **InfraTools Desktop Discovery connector**.

To learn how the **AssetCenter** connector works, refer to the Connect-It **User's guide**, chapter **Peregrine Systems connectors**, section **Asset Management connector**.

To learn how th **iddac.scn** scenario works and which data is recovered, refer to the Connect-It **Predefined scenarios** guide, chapter **InfraTools Desktop Discovery - Asset Management scenario**.

Verify null-identifier records module (History)

This module verifies integrity of the records whose primary keys are null.

These records are automatically created in all the tables when the database is created.

They are used by AssetCenter to perform certain administrative tasks (which is transparent to you).

This module verifies that these records still exists, and will recreate them if necessary.

We recommend that you execute this module at least once every day to maintain the integrity of the database.

Purge the input-events table module (PurgeEventInTable)

This module deletes the records of the **Input events** table (amInputEvent) according to the information in the:

- **Status** field (seStatus) of the **Input events** table (amInputEvent).
- **Deletion** field (seStatus) of the **Input events** table (amInputEvent).
- Expiration time defined by the **Events management/ Expiration time for input events (in hours)**, accessible via the **Tools/ Database options** menu item in the AssetCenter application.

Purge the outgoing-events table module (PurgeEventOutTable)

This module deletes the records from the **Input events** table according to the information in the:

- **Status** field (seStatus) of the **Output events** table (amOutputEvent).
- **Deletion** field (seStatus) of the **Output events** table (amOutputEvent).
- Expiration time defined by the **Events management/ Expiration time for output events (in hours)**, accessible via the **Tools/ Database options** menu item in the AssetCenter application.

Update statistics for tables module (Stats)

This module updates the database statistics.

These statistics are used by all the DBMSs supported by AssetCenter to optimize SQL query plans.

If these statistics are not updated, the DBMS will not know which indexes are the most efficient.

We recommend that you execute this module once a week, or every night if your database is heavily modified.

Activating the verification immediately

It is possible to activate a verification immediately, without having to wait for the defined period to elapse. You can do this using the **Action/ Activate** menu item under AssetCenter Server.

Indicate which verifications to carry out by checking the appropriate boxes.

Administrating AssetCenter Server by the Web

The installation program of AssetCenter under Windows NT installs AssetCenter Server as an NT service.

Thus, you have the choice between launching:

- The graphical interface of AssetCenter Server.
- Or AssetCenter Server as an NT service.

If you launch AssetCenter Server as an NT service, you can control its functioning via the Web.

This section deals with:

- Launching AssetCenter Server as an NT service.
- Accessing the AssetCenter Server service via the Web.
- Driving the AssetCenter Server service via the Web.

Launching AssetCenter Server as an NT service

During the installation of AssetCenter:

- AssetCenter Server is installed as an NT service that is not launched.
- Web access to the AssetCenter Server service is not enabled.

Note: Before installing the AssetCenter Server, we recommend creating a Windows NT user account on the machine on which this service will be installed. Next, we recommend installing the AssetCenter Server service under this account. (Warning: The account must have the necessary rights to start the AssetCenter Server service an establish a connection with the database engine.)

Enabling Web access to the AssetCenter Server service

To enable Web access:

- Edit **amsrvcf.ini**, which can be found in the **amsrv\bin32** subfolder of the AssetCenter installation folder.
- In the [GLOBAL] section, modify the value of the "WebAdmin" key:
  - If WebAdmin = 1, Web access is enabled.
  - If WebAdmin = 0, Web access is disabled.
- In the [GLOBAL] section, check the value of the TCP/IP port used by the AssetCenter Server service. This value is stored in the "WebPort" key and is by default equal to 82. Modify this value if this port is already used by another program.

Start the AssetCenter Server service

To start the AssetCenter Server NT service:

1. In Control Panel, click the **Services** utility.
2. Select the AssetCenter Server service.

Then, if you want to start the service straightaway:

- Click **Start**. In the case of the AssetCenter Server service, we advise against specifying any parameters directly in the **Startup Parameters** field.

If you want to configure the AssetCenter Server service:

1. Click the **Startup** button.
2. Specify whether you want the **Startup Type** to be:
   - **Automatic**: In this case, the service is launched when Windows NT is started.
   - **Manual**: In this case, the service must be launched manually by clicking **Start** in the Windows NT services utility.
   - **Disabled**: In this case, the NT service cannot be launched.

Accessing the AssetCenter Server service via the Web

Note: Warning: In order to be able to access the AssetCenter Server service via the Web, the service must be started.

Then, to access the AssetCenter Server service:

1. Launch your usual Web browser.
2. Enter the address of the computer on which the AssetCenter Server service is running, followed by the TCP/IP port number used by the AssetCenter Server service on this machine. The address of the computer and the port are separated by a colon ":".

   Example 12.1. Examples of addresses:

   - "http://colombo.taltek.com:82".
   - "http://laguardia.taltek.com:800".

It is also possible to enter the TCP/IP address of the computer on which the AssetCenter Server service is running, followed by the value of the port.

Example 12.2. Example:

"127.0.0.1:82".

3 The Home Page is opened. Click the **Connect** button in this page.

4 A window authorizing access to the AssetCenter Server service is displayed. Use it to enter:

    1 A "UserName": "WebAdmin".

    2 The password associated with the "WebAdmin" UserName. By default, this password is empty.

Driving the AssetCenter Server service via the Web

This section describes the commands you have at your disposal once connected to the AssetCenter Server service.

Connect to new database

Using this command, you can:

- Connect manually to an AssetCenter database.
- Configure AssetCenter Server to connect automatically to an AssetCenter database at startup. To do this:

    1 Check the **Reconnect at startup** option.

    2 Enter the name of the database connection to which the AssetCenter Server service must reconnect automatically.

    3 Specify the **Login** name and password.

Note: You can also configure the automatic connection of the AssetCenter Server service via the "AutoLogin" key in the "Database" section of the amsrvcf.ini file. AutoLogin = 0:

Automatic connection disabled. AutoLogin = 1: Automatic connection is enabled.

Status of the server

This command displays the last 100 messages from the AssetCenter Server log file. These messages are similar to those shown in the main window in the GUI version of AssetCenter Server.

Click **Clear** to clear the all the messages displayed.

Note: The maximum number of messages cannot be modified.

Configure

Using this command, you can define which modules are to be verified.

Note: You cannot modify the verification schedules concerning the modules via the Web. To do this, you must use the Options/Configure menu item in AssetCenter Server's graphical interface.

Activate

Select this command to activate certain verifications straightaway.

WebAdmin password

Use this command to modify the "WebAdmin" password.

By default, this password is empty.

Exit

Click this command to disconnect from the AssetCenter Server service.

Note: There is an automatic disconnection option in case of inactivity. This option is defined by the "TimeOut" key in the [SESSION] section of the amsrvcf.ini file. By default, it is set to 10 minutes.

13 Managing messages

CHAPTER

AssetCenter gives you the possibility of managing two types of messages:

- Messages issued from AssetCenter and sent to the AssetCenter database via its internal messaging system.
- Messages created in AssetCenter and sent via an external messaging system.

Overview of messaging

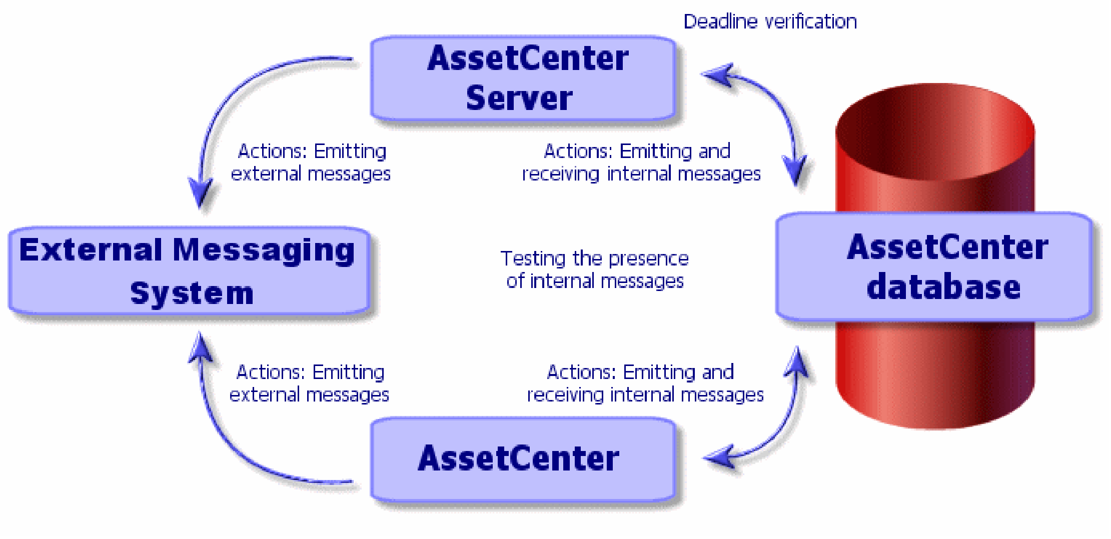
AssetCenter manages message sending by using the following protocol types:

- AM (AssetCenter)
- SMTP
- MAPI
- VIM

When receiving messages, AssetCenter only manages those messages using the AM (AssetCenter) protocol.

Overview

Figure 13.1. Overview of internal messaging system


Deadline verification
AssetCenter Server
Actions: Emitting external messages
Actions: Emitting and receiving internal messages
External Messaging System
Testing the presence of internal messages
AssetCenter database
Actions: Emitting external messages
Actions: Emitting and receiving internal messages
AssetCenter

How to issue messages

Messages are generated via **Messaging** actions.

These actions must first be created in order for a message to be able to be sent.

The action is triggered in different ways:

- Manually, by selecting the action in the list given by the **Tools/ Actions** menu item.
- Automatically, via AssetCenter Server.
- Automatically, via AssetCenter.

The creation of "Messaging" type actions is described in the "Administration" guide, chapter "Defining actions", section "Creating an action", sub-section "Populating the Messaging tab.

How to consult messages

Consulting messages sent to the internal messaging system

An agent test for the arrival of new internal messages and informs the users of the presence of new messages.

These messages can be consulted:

- Using the **Tools/ Messages** menu item.
- From the message box which informs of the presence of new messages.

Consulting messages sent to an external messaging system

The recipient of these messages can consult them in the usual way under the external messaging system client.

Acknowledgement of receipt

Note: You cannot receive an acknowledgement for a message sent via the AssetCenter internal messaging system, or via a MAPI or SMTP messaging system.

To receive acknowledgment, check the **Acknowledgment** box (SQL name: bAcknowledgment) field in the detail of the **Messaging** type action.

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 AssetCenter database (in the Departments and Employees table).

Configuring AssetCenter to use messaging systems

The configuration of AssetCenter depends of the type of protocols used.

In order for AssetCenter and AssetCenter Server to be able to send messages to an external messaging system, you need to:

- Populate certain fields in the employee detail screens.
- Populate certain fields of an action.
- Configure and activate AssetCenter Server.
- Configure the frequency of testing for new messages.

  Using the **Tools/ Options** menu item, **Messaging** tab.

Warning: When you use AssetCenter, you cannot use another messaging protocol other than the one already used.

Multiple recipients

Whatever protocol you use, you have to respect the following syntax in order to send a message to several recipients:

```
SMTP:[name@addresse.domain],
SMTP:[name2@address.domain]
```

For example:

```
SMTP:[andreoucassidou@peregrine.com],
SMTP:[administrator@prgn.com]
```

SMTP protocol

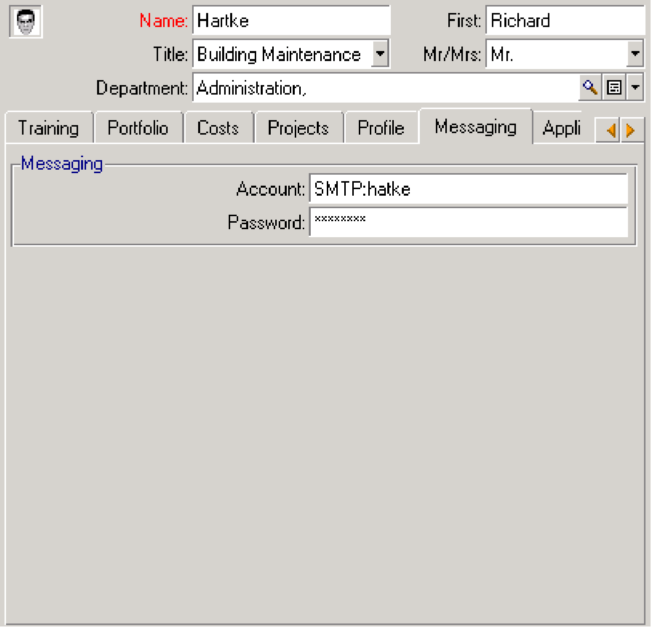
Detail of the user

In order for a message to be able to be sent, AssetCenter needs to know the account of the sender, specified in the Departments and Employees table (**Messaging** tab), and the messaging address of the recipient, specified in an action.

- The account of the sender is identified by the following fields:
  - **Account** (SQL name: MailLogin): this must be in the form:

```
SMTP:[name]
```

- **Password** (SQL name: MailPassword): Leave this field empty, unless your SMTP server requires a login.

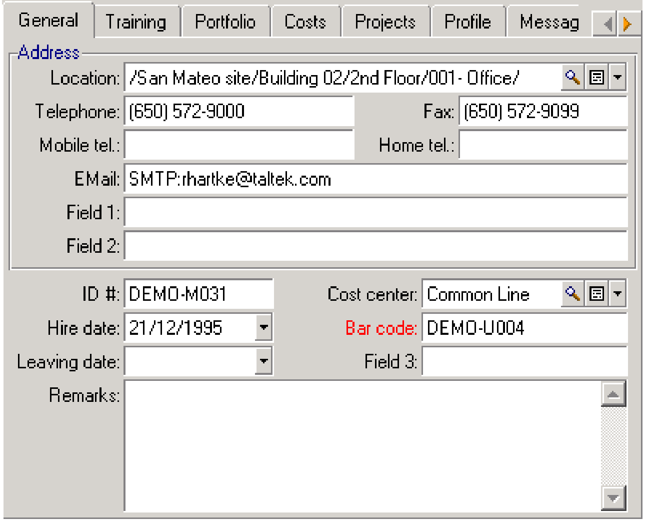


Name: Hartke
First: Richard
Title: Building Maintenance
Mr/Mrs: Mr.
Department: Administration,

Messaging
Account: SMTP:hatke
Password: ********

- Populate the **Email** field of the **General** tab with the following syntax:

```
SMTP:[name@address.domain]
```



Address
Location: /San Mateo site/Building 02/2nd Floor/001 - Office/
Telephone: (650) 572-9000
Fax: (650) 572-9099
Mobile tel.:
Home tel.:
EMail: SMTP:rhartke@taltek.com
Field 1:
Field 2:

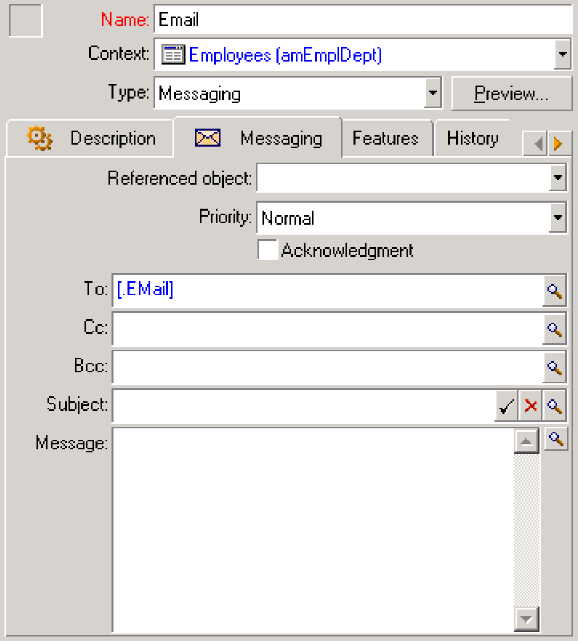
ID #: DEMO-M031
Cost center: Common Line
Hire date: 21/12/1995
Bar code: DEMO-U004
Leaving date:
Field 3:
Remarks:

- The recipient is identified by the **To** field (SQL name: MsgTo) in the detail of a messaging-type action.

  Populate the **To** field (and the optional **Cc** and **Ccc** fields, if you wish) with the appropriate address, in the following manner:

  ```
  SMTP:[name@address.domain]
  ```

  Or using a calculated string.



Name: Email
Context: Employees (amEmplDept)
Type: Messaging
Referenced object:
Priority: Normal
□ Acknowledgment
To: [.EMail]
Cc:
Bcc:
Subject:
Message:

In this example, the **To** field is populated with the contents of the **Email** field, in the **General** tab of a user's detail. The administrator is put in copy.

Win.ini file

You must add certain command lines in the **win.ini** file of your Windows folder in order to be able to use the messaging system in AssetCenter.

```
[mail]
SMTP=1
SMTPserver=[server name]
```

```
email=[messaging address@domain name]
displayname=[user's full name]
```

Example

```
[mail]
SMTP=1
SMTPserver=mail.prgn.com
email=sblaine@prgn.com
displayname=Steven Blaine
```

MAPI protocol

Detail of the user

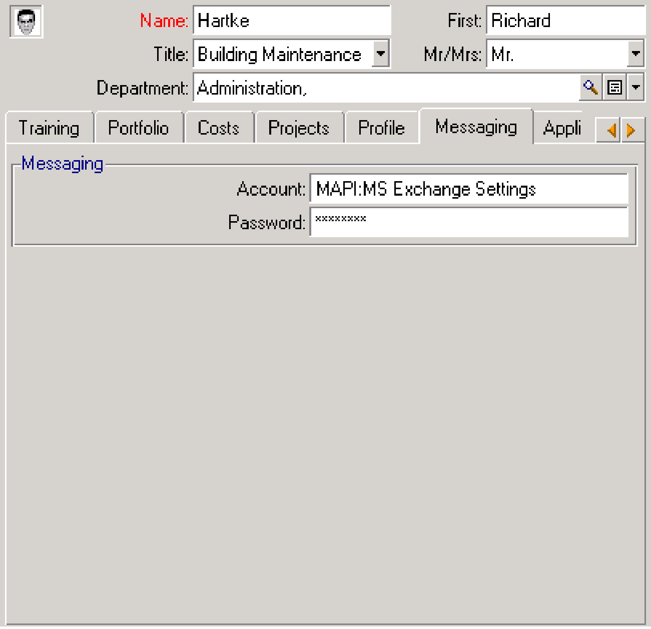
In order for a message to be able to be sent, AssetCenter needs to know the account of the sender, specified in the Departments and Employees table (**Messaging** tab), and the messaging address of the recipient, specified in an action.

- The profile of the sender is identified by the following fields:
  - **Account** (SQL name: MailLogin): Populate the field in the following manner:

    ```
    MAPI:[user profile name]
    ```

    To find out the name of the user profile, look in the Windows **Configuration Panel** and click the **Mail** icon. In the window that appears, click **Show Profiles**.

- **Password:** (SQL name: MailPassword): Populate the field with your messaging password.



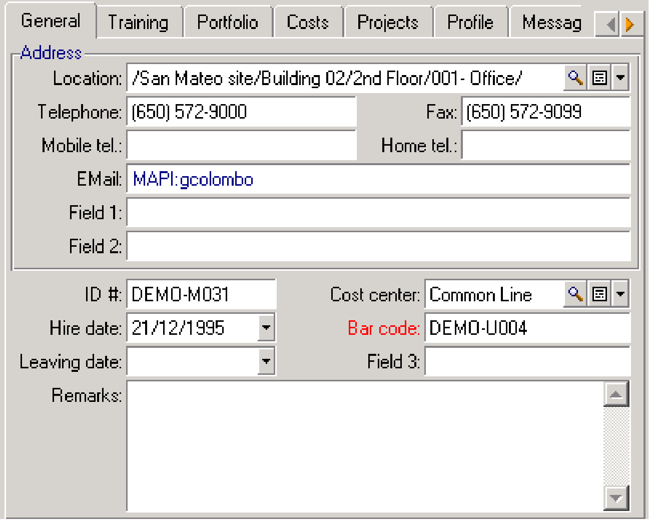
Name: Hartke
First: Richard
Title: Building Maintenance
Mr/Mrs: Mr.
Department: Administration,

Messaging

Account: MAPI:MS Exchange Settings
Password: ********

- Populate the **Email** field of the **General** tab with the following syntax:

```
MAPI:[mailbox name]
```



Address

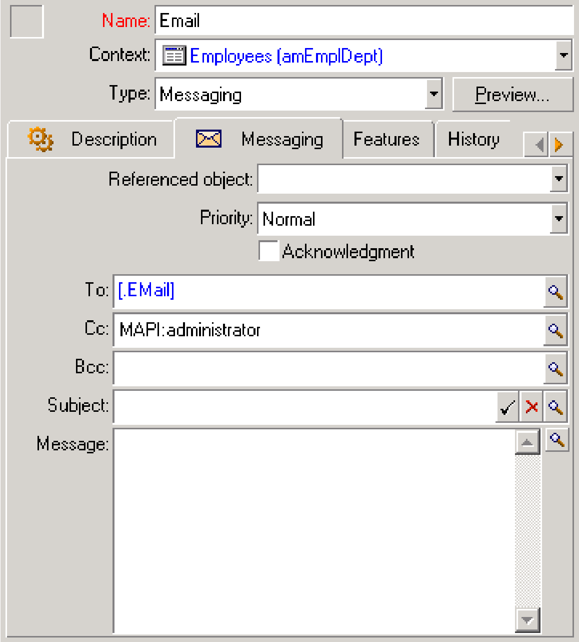
Location: /San Mateo site/Building 02/2nd Floor/001- Office/
Telephone: (650) 572-9000
Fax: (650) 572-9099
Mobile tel.:
Home tel.:
EMail: MAPI:gcolombo
Field 1:
Field 2:
ID #: DEMO-M031
Cost center: Common Line
Hire date: 21/12/1995
Bar code: DEMO-U004
Leaving date:
Field 3:
Remarks:

- The recipient is identified by the **To** field (SQL name: MsgTo) in the detail of a messaging-type action.

  Populate the **To** field (and the optional **Cc** and **Ccc** fields, if you wish) with the appropriate address, in the following manner:

  ```
  MAPI:[mailbox name]
  ```

  Or using a calculated string.



Name: Email
Context: Employees (amEmplDept)
Type: Messaging
Referenced object:
Priority: Normal
□ Acknowledgment
To: [.EMail]
Cc: MAPI:administrator
Bcc:
Subject:
Message:

In this example, the **To** field is populated with the contents of the **Email** field, in the **General** tab of a user's detail.

VIM protocol

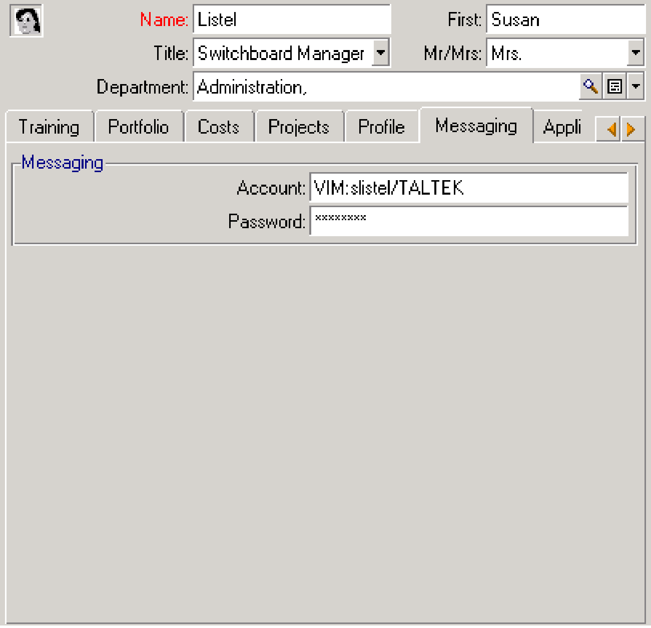
Note: AssetCenter does not manage accented characters for the sender's identifier with the VIM protocol.

Detail of the user

In order for a message to be able to be sent, AssetCenter needs to know the account of the sender, specified in the Departments and Employees table (**Messaging** tab), and the messaging address of the recipient, specified in an action.

- The account of the sender is identified by the following fields:
  - **Account** (SQL name: MailLogin): this must be in the form:

```
VIM:[name/domain]
```

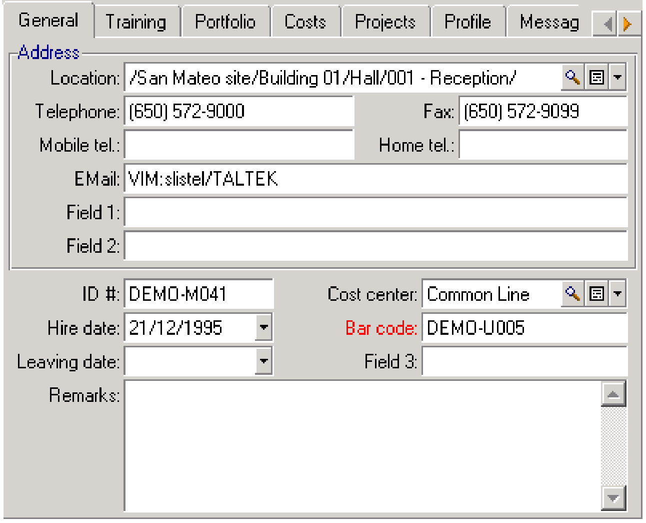


Name: Listel
First: Susan
Title: Switchboard Manager
Mr/Mrs: Mrs.
Department: Administration,

Account: VIM:slistel/TALTEK
Password: ********

  - **Password**: (SQL name: MailPassword): Populate the field with your messaging password.
- Populate the **Email** field (SQL name: Email) of the **General** tab with the following syntax:

```
VIM:[name/domain]
```



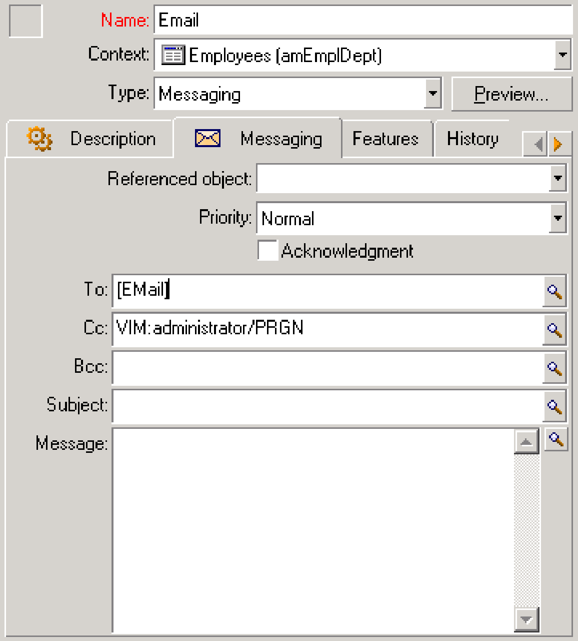
Address

Location:	/San Mateo site/Building 01/Hall/001 - Reception/		
Telephone:	(650) 572-9000	Fax:	(650) 572-9099
Mobile tel.:		Home tel.:	
EMail:	VIM:slistel/TALTEK		
Field 1:			
Field 2:			
ID #:	DEMO-M041	Cost center:	Common Line
Hire date:	21/12/1995	Bar code:	DEMO-U005
Leaving date:		Field 3:	
Remarks:			

- The recipient is identified by the **To** field in the detail of a messaging-type action.

  Populate the **To** field (and the optional **Cc** and **Ccc** fields, if you wish) according to the following syntax:

```
VIM:[name/domain]
```



Name: Email

Context: Employees (amEmplDept)

Type: Messaging

Referenced object:

Priority: Normal

□ Acknowledgment

To: [EMail]

Cc: VIM:administrator/PRGN

Bcc:

Subject:

Message:

In this example, the **To** field is populated with the contents of the **Email** field, in the **General** tab of a user's detail. The administrator is put in copy.

Windows configuration

You must configure the Windows **Path** in order for AssetCenter to be able to manage the VIM protocol.

Windows 2000

- Note the access path to the **vim32.dll**.
- Edit the properties of your computer by right-clicking the icon of your workstation (system).
- In the **Advanced** tab, click **Environment variables**.
- In the **System variables**, edit the **Path** variable by adding the Lotus Notes path.

Windows NT4

- Note the access path to the **vim32.dll** file of your Lotus directory.
- Edit the properties of your computer by right-clicking the icon of your workstation (system).
- In the **Environment** tab, edit the **Path** variable by adding the Lotus Notes path.

Windows 95, 98, ME

- Note the access path to the **vim32.dll** file of your Lotus directory.
- Locate and edit the autoexec.bat file at the root of your hard drive.
- Add the instruction

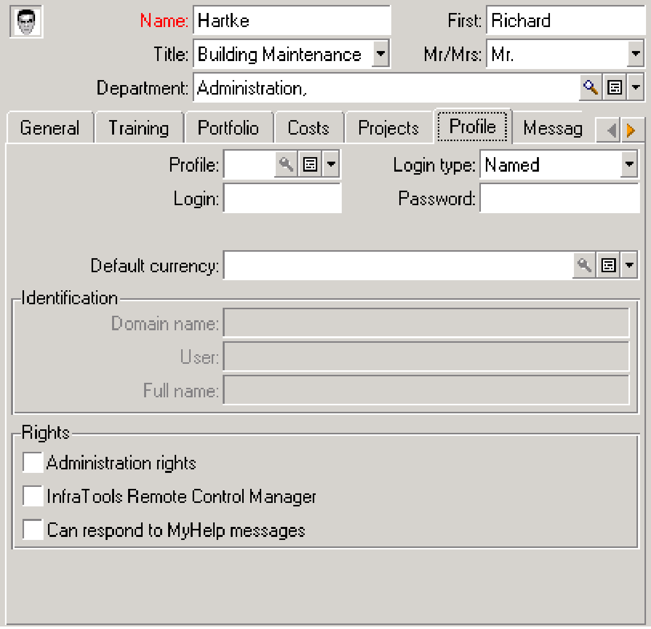
  ```
  SET PATH=[Lotus Notes path]
  ```

  and specify the access path to the **vim32** of your Lotus Notes directory.

AM protocol

- In order for an internal message to be sent, the user must be registered in the Departments and Employees table (amEmplDept) and have a login:

- The login of the user is specified in the Departments and Employees table (**Profile** tab).



Name: Hartke
First: Richard
Title: Building Maintenance
Mr/Mrs: Mr.
Department: Administration,

Profile:
Login type: Named
Login:
Password:
Default currency:

Identification
Domain name:
User:
Full name:

Rights
□ Administration rights
□ InfraTools Remote Control Manager
□ Can respond to MyHelp messages

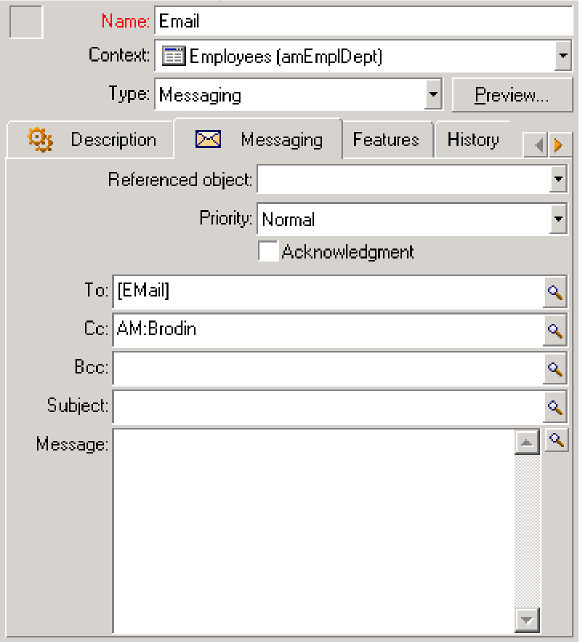
- You must also populate the **Email** field in the **General** tab of the user with the following syntax:

```
AM:[user login]
```

- The messaging address of the recipient, specified in a messaging-type action, must be entered:
  - Populate the **To** field (and the optional **Cc** and **Ccc** fields, if you wish) in the following manner:

```
AM:[login of recipient]
```

Or using a calculated string.



Name: Email

Context: Employees (amEmplDept)

Type: Messaging

Referenced object:

Priority: Normal

□ Acknowledgment

To: [EMail]

Cc: AM:Brodin

Bcc:

Subject:

Message:

In this example, the **To** field is populated with the contents of the **Email** field, in the **General** tab of a user's detail.

Common connection problems

When a problem is encountered in sending a message, the administrator is sent a message to notify them of the problem.

Test to validate a connection to a messaging system

1. Launch AssetCenter Server.
2. Connect to a database.
3. Click ✉.

Test to perform if a problem is encountered

1. Create a new **Messaging Type** (SQL name: seActionType) action specifying a given recipient.
2. Trigger the action using the **Tools/ Actions** menu item.
3. Verify that the recipient has received the message and that the "router" has not sent an error message to your messaging system (unknown recipient).
4. Consult any error messages.

Troubleshooting

"Connection to messaging system 'XXX': no messaging system specified. Verify the prefix of the messaging account in the 'Profile' tab in the employee detail."

You need to prefix the **Account** field (SQL name: MailLogin) in the **Messaging** tab in the employee detail by:

- "MAPI:" if you use a MAPI standard messaging system (Microsoft Outlook, Microsoft Exchange, etc.)
- "VIM:" if you use a VIM standard messaging system (Lotus Notes, CCMail, etc.)
- "SMTP:" if you use an SMTP standard messaging system (Internet standard)

"Unable to connect to messaging 'XXX'."

The **Account** field in the **Messaging** tab of the employee detail is correctly prefixed by "MAPI:" or "VIM:", but the name of the account is incorrect. Verify that this has been correctly entered.

"Messaging account 'VIM': a password must be given (this cannot be empty).

If you use a VIM messaging system, you must specify a password in the **Password** field (SQL name: MailPassword) in the **Messaging** tab of the employee detail. The password cannot be empty.

"Messaging account 'XXX': incorrect password."

The password specified in the **Password** field in the **Messaging** tab of the employee detail is incorrect.

"Message not sent to 'XXX': messaging not available."

This reveals a problem in the "win.ini" file.

In order for AssetCenter to work correctly with MAPI messaging systems, the "win.ini" file must contain the following lines in the [Mail] section:

MAPI=1

MAPIX=1

In order for AssetCenter to work correctly with VIM messaging systems, the "win.ini" file must contain the following line in the [Mail] section:

SMI=1

In order for AssetCenter to work correctly with SMTP messaging systems, the "win.ini" file must contain the following lines in the [Mail] section:

SMTP=1

SMTPServer=<Outgoing mail server>

The following lines are optional:

SMTPPort=<Port number of outgoing mail server> (25 by default)

SMTPTimeOut=<Timeout value> (20 seconds by default)

These four lines are not exclusive.

If one of these lines is not present or its value is set to 0, you need to verify the proper functioning of the corresponding messaging system. To do this, use a program such as Microsoft Internet Mail for MAPI and Lotus Notes for VIM. If the messaging is working and you are not in the case described below, you can modify the [Mail] section of the "win.ini" file as shown above.

Warning: If MAPI is set to 1 but not MAPIX, it is possible that the messaging system does not support extended MAPI. Verify whether this is the case. AssetCenter cannot function correctly is the messaging system is not compatible with extended MAPI.

"Error opening VIM session: password required"

With "VIM" messaging systems a password is required. Add it under your messaging system and indicate it in AssetCenter in the **Password** field (SQL name: MailPassword) in the **Messaging** tab of the employee detail.

"Error opening VIM session: Incorrect password"

The password is invalid. Modify the value of the **Password** field (SQL name: MailPassword) in the **Messaging** tab of the employee detail.

"Message container is corrupt" or **"The configuration of your station is invalid"**.

The VIM protocol recovers the information linked to the name and the location of the Lotus Notes **notes.ini** file. If this information is incorrect, you cannot send messages. Edit and modify the parameters in this file accordingly.

14 Workflow

CHAPTER

This chapter describes in detail how to define and manage workflow schemes.

Definitions

This section defines several key terms used in workflow:

- Workflow
- Workflow activity
- Workflow event
- Workflow transition
- Workflow task
- Workflow activity assignee
- Definition of a workflow execution group

Workflow

Workflow is concerned with the formalization and/or automation of business procedures.

For example, the following processes can be modeled and automated using workflow methods:

- Purchase-request approval procedures.
- Asset moves.
- Etc.

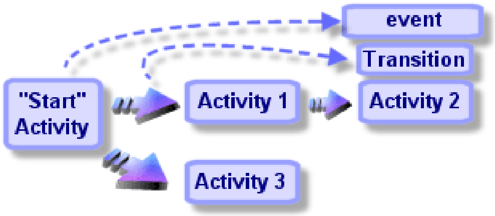
AssetCenter makes it possible for you to define workflow schemes and manage their progress.

Workflow scheme

Creating a workflow scheme in AssetCenter consists of defining:

- Activities.
- Trigger events (events resulting from activities that make it possible to activate transitions).
- Trigger transitions (transitions that trigger activities).
- A context.
- Time limits and alarms.

Figure 14.1. Simplified representation of a workflow scheme



Use the Tools/ Workflow/ Workflow schemes menu item to access the table of workflow schemes (SQL name "amWfScheme").

Workflow instance

In this document, a "Workflow instance" refers to a defined workflow scheme that is being executed.

Workflow activity

A workflow consists of:

- A task to be executed. This task may necessitate user interaction or be carried out automatically by AssetCenter Server.
- Events that trigger transitions to other activities.

Workflow activities are stored in the table of workflow activities (SQL name: amWfActivity).

Use the graphical editor in the Activities tab of a workflow detail to access the detail of a workflow activity.

Workflow event

Workflow events results from activities. They in turn make it possible to activate transitions that trigger other activities.

Events belonging to a workflow scheme are stored in the table of workflow events (SQL name: amWfEvent).

When these events occur, they are recorded in the table of elementary events of a workflow instance (SQL name: amWfOccurEvent).

Use the graphical editor in the Activities tab of a workflow detail to access the detail of workflow event.

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.

Transitions that belong to a workflow scheme are stored in the table of workflow transitions (SQL name: amWfTransition).

Use the graphical editor in the Activities tab of a workflow detail to access the detail of workflow transitions.

Workflow task

A workflow task is an assigned task to be carried out, resulting from the triggering of an activity.

In order for workflow tasks to be recorded in the table of workflow tasks (SQL name: amWfWorkItem), the **Log task** option (SQL name:bLogWorkItem), in the **General** tab of the activity detail must be checked.

Use the Tools/ Tasks in progress menu item to access the list of workflow tasks to be carried out.

Workflow activity assignee

Assignees are appointed to perform tasks resulting from **Question** or **User action** type workflow activities. Assignees are not involved in **Automatic action** or **Test / script** type activities.

Workflow assignees are stored in the table of workflow roles (SQL name: amWfOrgRole). Use the Tools/ Workflow/Roles menu item to access the table of workflow roles.

Definition of a workflow execution group

Workflow execution groups enable you to classify the workflow schemes that you define. The execution group to which a given workflow scheme belongs is indicated in the **Execution group** field (SQL name: GroupName) in the **General** tab of the workflow detail.

General overview

The first step in implementing workflow under AssetCenter is to define workflow schemes using the graphical editor via the **Tools/ Workflow/ Workflow schemes** menu item. Workflow schemes define activities, events and transitions. They can reference AssetCenter actions and employees (workflow assignees).

Workflow schemes are interpreted by workflow engines. The workflow engines in question are run by either AssetCenter Server, or agents of AssetCenter.

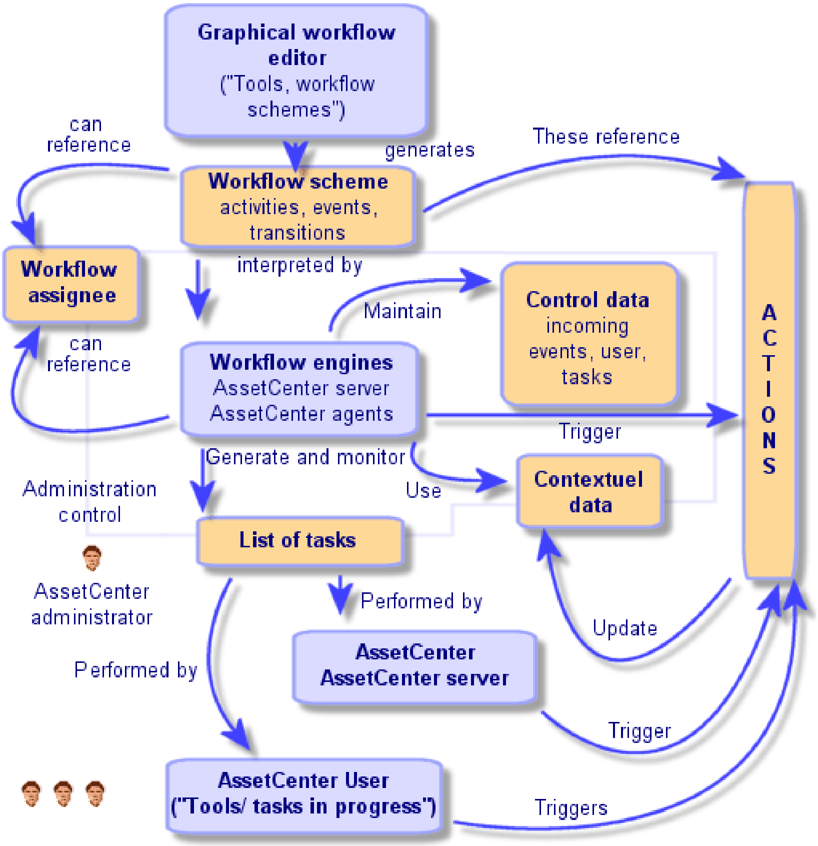
In reaction to events, workflow engines trigger and monitor workflow instances:

- 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 AssetCenter 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 AssetCenter:

Figure 14.2. Overview of workflow in AssetCenter

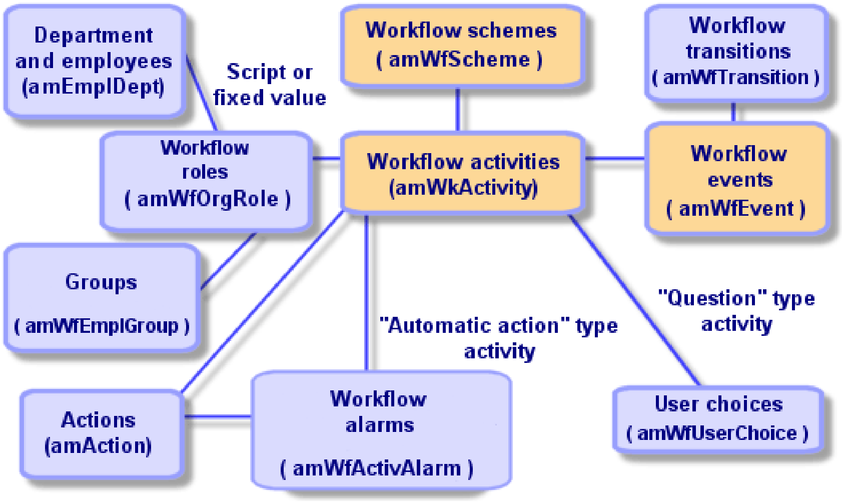

Graphical workflow editor ("Tools, workflow schemes")
generates
Workflow scheme activities, events, transitions
can reference
Workflow assignee
These reference
interpreted by
Maintain
Control data incoming events, user, tasks
Workflow engines AssetCenter server AssetCenter agents
can reference
Trigger
ACTIONS
Generate and monitor
Use
Contextuel data
Administration control
List of tasks
AssetCenter administrator
Performed by
Update
AssetCenter AssetCenter server
Performed by
Trigger
AssetCenter User ("Tools/ tasks in progress")
Triggers

The processing of workflow instances varies according to the nature of activities and events and the way in which the workflow engines controlling them have been configured.

Main tables involved in workflow

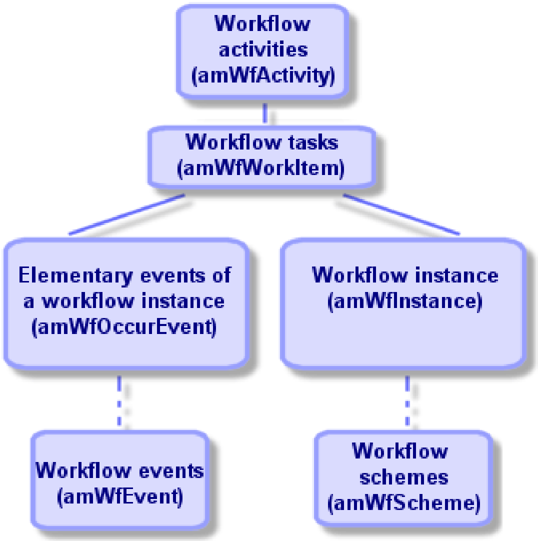
The following diagram presents the main tables involved in workflow and the links between them. The tables are identified by their label and SQL name:

Figure 14.3. Main tables allowing you to define a workflow scheme


Department and employees (amEmplDept)
Script or fixed value
Workflow schemes (amWfScheme)
Workflow transitions (amWfTransition)
Workflow roles (amWfOrgRole)
Workflow activities (amWkActivity)
Workflow events (amWfEvent)
Groups (amWfEmplGroup)
"Question" type activity
"Automatic action" type activity
Actions (amAction)
Workflow alarms (amWfActivAlarm)
User choices (amWfUserChoice)

The following diagram presents the main tables involved when a workflow instance is in progress:

Figure 14.4. Main tables involved in a workflow instance being executed


Workflow activities (amWfActivity)
Workflow tasks (amWfWorkItem)
Elementary events of a workflow instance (amWfOccurEvent)
Workflow instance (amWfInstance)
Workflow events (amWfEvent)
Workflow schemes (amWfScheme)

Using the graphical workflow editor

Use the Tools/ Workflow/ Workflow schemes menu item to access the workflow schemes. This menu item is reserved for the AssetCenter administrator. Use the Tools/ Workflow/ Workflow schemes menu item to access the workflow schemes. This menu item is reserved for AssetCenter 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 in order to create, modify or delete workflow schemes:

- Activities
- Events
- Transitions
- Other functionality

Activities

To create an activity:

- Either right-click an empty area of the **Activities** tab, then select the **Add an activity** command from the shortcut menu. The detail of the activity is displayed.
- Or alternatively, click the ▤ button, then click in the graphical zone. The detail of the activity is displayed.

To delete an activity:

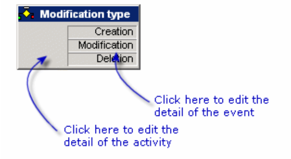
- Either select the activity by clicking it then press the "Delete" key.
- Or select the activity and select the **Delete** command from the shortcut menu.
- Or select the activity and click the ✕ button.

To modify the detail of an activity:

- Either right-click the activity and select the **Detail of activity** item from the shortcut menu.

- Or double-click the activity.

Figure 14.5. Detail of an activity



Modification type
Creation
Modification
Deletion
Click here to edit the detail of the event
Click here to edit the detail of the activity

Events

Database or Periodical type event

To add an output event to an activity:

- Either right-click the activity, then select the **Add event** command from the shortcut menu.
- Or 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 double-click the event and select the **Delete** command from the shortcut menu.
- Or select the event and click the button.

To modify the detail of a **Database** or **Periodical** type event:

- Either double-click the event.
- Or select the event then select the **Detail of event** item from the shortcut menu.

System event

You can create and delete **System** events from the activities at their origin.

To modify a **System** event, use one of the following methods as appropriate:

- To modify the processing method of an event (**Processing** field (SQL name: seProcessingMode) in the detail of the event), proceed as if modifying a **Database** or **Periodical** type event.
- Else, edit and modify the detail of the activity at the origin of the event.

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 select the **Delete** command from the shortcut menu.
- Or select the transition and click the × button.

To modify the detail of a transition:

1. Click the transition to select it.
2. Select the **Detail of transition** item from the shortcut menu.

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.

How to implement workflow

Workflow under AssetCenter makes it possible to automate the procedures of your company. Here are the steps to follow:

1. Analyze the procedures of your company that you want to formalize.
2. Create:
   1. Workflow roles.
   2. Actions.
3. Create workflow schemes. Define:
   1. Activities, events and transitions.
   2. Alarms.
4. Define the appropriate workflow execution groups. Associate each workflow scheme with a workflow execution group.
5. Launch AssetCenter Server on one or more machines. For each instance of AssetCenter Server, define the workflow execution groups to be monitored and the monitoring parameters.

Example of workflow used in request approval

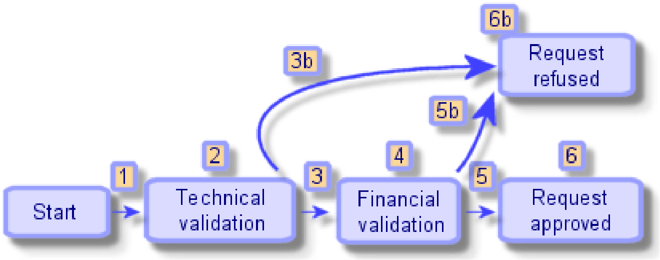
This section details a simple example of workflow.

1. Aim
2. Preliminary steps
3. Creating activities
4. Creating the start event
5. Creating transitions
6. Example of activating a workflow instance

Aim

The aim of this workflow scheme is to automate the purchase request process according to the following:

Figure 14.6. Workflow scheme



The steps of the workflow scheme are as follows:

1 The workflow instance starts as soon a purchase request is to be validated, i.e. when the **Request status** field (SQL name: seApprStatus) the purchase request detail indicates **Awaiting approval**.

2 The request first undergoes technical validation. 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, the next step is financial validation.

Otherwise the request is refused.

4 Financial validation consists of submitting the request to the company's financial controller, Mr. Gerald Colombo. They are 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.

Otherwise the request is refused.

6 If the purchase request is approved, AssetCenter sets the **Request status** field (SQL name: seApprStatus) in the purchase request detail to **Validated**.

When the purchase request is refused, AssetCenter sets the **Request status** field (SQL name: seApprStatus) in the purchase request detail to **Refused**.

In both cases, AssetCenter notifies the requester.

Preliminary steps

You must connect to the database under the login "Admin" and configure the messaging system. (See chapter 13 in this guide.)

Creating workflow assignees

The assignees involved in this workflow scheme are:

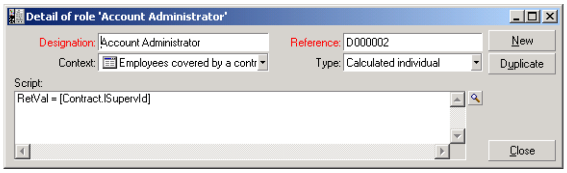
- The departmental supervisor of the requester.
- The financial controller of the company, Mr. Gerald Colombo.

The workflow assignees are stored in the table of workflow roles (SQL name: amWfOrgRole). To create them, select the **Tools/ Workflow/Roles** 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:

Figure 14.7. Example of a workflow



Detail of role 'Account Administrator'

Designation: Account Administrator

Reference: D000002

Context: Employees covered by a contr

Type: Calculated individual

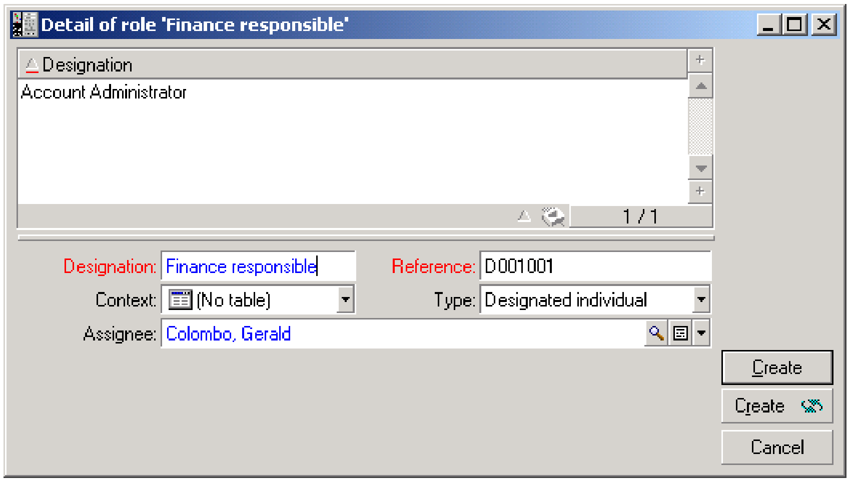
Script:

```
RetVal = [Contract.lSupervId]
```

Financial controller of the company

This person is designated as Mr. Gerald Colombo. To define him as an assignee, populate the detail screen as follows:

Figure 14.8. Detail of a workflow



Detail of role 'Finance responsible'

Designation
Account Administrator

1 / 1

Designation: Finance responsible

Reference: D001001

Context: (No table)

Type: Designated individual

Assignee: Colombo, Gerald

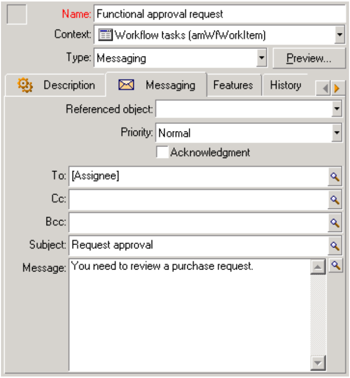
Creating actions

The workflow scheme calls on numerous actions. To create them, select the **Tools/ Actions/ Edit** 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:

Figure 14.9. "Request for technical validation" action



Name: Functional approval request

Context: Workflow tasks (amWfWorkItem)

Type: Messaging

Referenced object:

Priority: Normal

□ Acknowledgment

To: [Assignee]

Cc:

Bcc:

Subject: Request approval

Message: You need to review a purchase request.

Request for financial validation sent to financial controller

The "Request for financial validation" action is used in the financial approval process. It is similar to the "Request for function validation" action described above. It sends a message to the person in charge of financial validation of the need to review this request:

Validation of the purchase request

This action is used at the level of the "Request approved" activity.

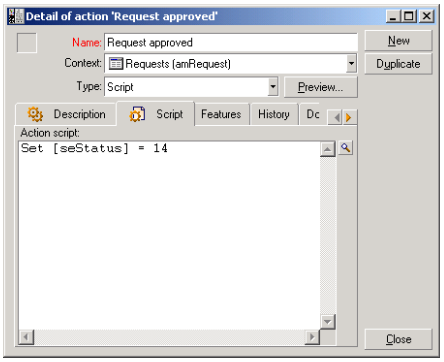
It sets the **Request status** field (SQL name: seApprStatus) in the purchase request detail to **Validated**. This action is a **Script** type action.

The **Request status** field is a system itemized list. To see the list of its values, display the help on this field:

1. Right-click the **Request status** field in the detail of the request.
2. Select the **Help on this field** menu item from the shortcut menu: The value displayed as **Approved** is stored in the database as "1".

The action is as follows:

Figure 14.10. "Request approved" action



Name: Request approved

Context: Requests (amRequest)

Type: Script

Action script:

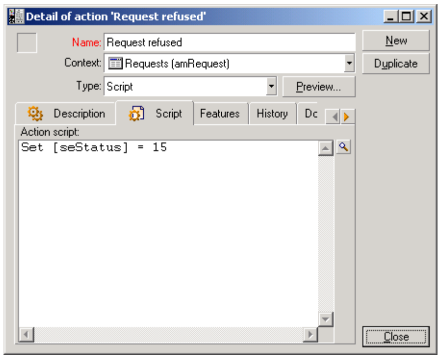
```
Set [seStatus] = 14
```

Refusal of the purchase request

This action is used at the level of the "Request refused" activity.

It is similar to the "Request approved" action, but the **Approval status** field (SQL name: seApprStatus) in the **General** tab of the purchase request detail must be set to **Refused**.

Figure 14.11. "Request refused" action



Name: Request refused

Context: Requests (amRequest)

Type: Script

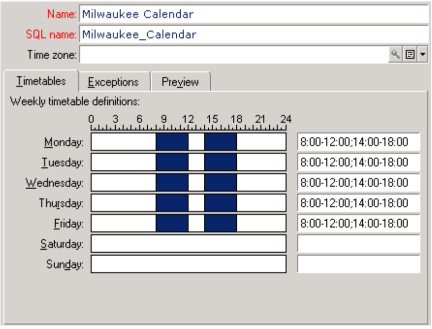
Action script:

```
Set [seStatus] = 15
```

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:

Figure 14.12. "Milwaukee Calendar" calendar



Name: Milwaukee Calendar
SQL name: Milwaukee_Calendar
Time zone:

Timetables | Exceptions | Preview

Weekly timetable definitions:

Monday:	8:00-12:00;14:00-18:00
Tuesday:	8:00-12:00;14:00-18:00
Wednesday:	8:00-12:00;14:00-18:00
Thursday:	8:00-12:00;14:00-18:00
Friday:	8:00-12:00;14:00-18:00
Saturday:	
Sunday:	

Preparing the workflow scheme

1 Launch the **Tools/ Workflow/ Workflow schemes** menu item.

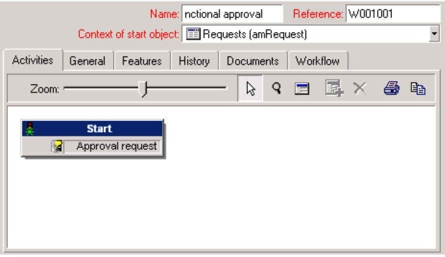
2 Click New.

3 Give a name to the workflow scheme.

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 AssetCenter in the graphical editor in the **Activities** tab.

Figure 14.13. "Request approval" workflow scheme



Name: nctional approval
Reference: W001001
Context of start object: Requests (amRequest)

Activities | General | Features | History | Documents | Workflow

Creating activities

Activities can be created graphically in the **Activities** tab of the workflow detail:

1. Click outside of a workflow object.
2. Right-click.
3. Select the **Add activity** menu item from the shortcut menu: The detail of the activity is displayed.

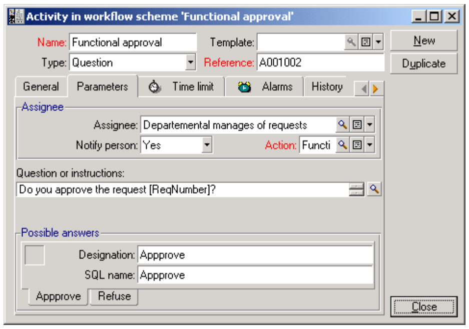
Technical validation

1. Assign a name to the 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.

Task to be carried out

1. Populate the **Parameters** tab as shown below:

Figure 14.14. Detail of a workflow activity



Activity in workflow scheme 'Functional approval'

Name: Functional approval

Template:

Type: Question

Reference: A001002

Assignee

Assignee: Departemental manages of requests

Notify person: Yes

Action: Functi

Question or instructions:

Do you approve the request [ReqNumber]?

Possible answers

Designation: Appprove

SQL name: Appprove

2. Specify the question to be asked:
   1. The text of the question references the number of the purchase request.
   2. 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** or **Duplicate link**.

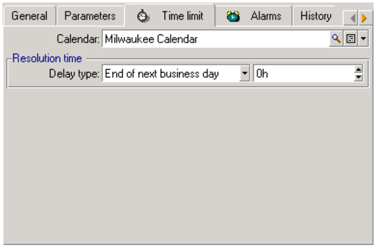
3. Indicate to whom the question is addressed in the **Assigned** field (SQL name: Assignee). In our case, the assignee is the departmental manager of the requester. This person was created in the table of workflow roles in the preliminary steps.
4. In order for the assignee to be notified of the need to review this request:
    1. Set the **Notify person** field (SQL name: bNotifAssignee) to **Yes**.
    2. Specify the action to be carried out: This is the "Request for technical approval" action created in the intermediary step. This action is automatically triggered when a purchase request is submitted to technical approval.

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 before end of the next business day.

Figure 14.15. Time limit of "Functional approval" activity



Calendar: Milwaukee Calendar

Resolution time

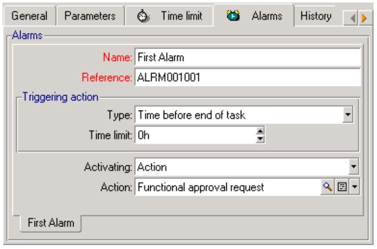
Delay type: End of next business day 0h

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 approval" action:

Figure 14.16. "Approval not carried out" alarm



Name: First Alarm
Reference: ALRM001001
Type: Time before end of task
Time limit: 0h
Activating: Action
Action: Functional approval request

It is possible to define further alarms using the **Add linked record** command from the shortcut menu.

Events

Once the activity is created, AssetCenter creates two system events **Approve** and **Refuse** corresponding to the two possible responses to the question.

When these events occur, an AssetCenter agent logs them in the table of workflow elementary events (SQL name "amWfOccurEvent"). AssetCenter Server takes care of triggering the following activities (by default, the **Processing** field (SQL name: seProcessingMode) in the **General** tab of the event detail is set to **Log event and process by server**).

Financial validation

This activity is similar to the previous one.

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. In order to notify him, select the "Request for financial validation" action.

Request approved

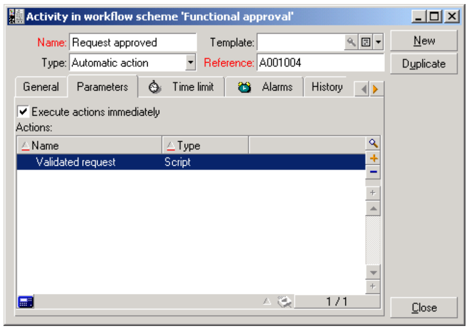
When the request succeeds in passing the two validation steps, it is approved.

The "Request approved" activity is one of the possible endings of the workflow scheme.

This activity needs to modify the detail of the request in order 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.

Figure 14.17. "Request approved" activity



Activity in workflow scheme 'Functional approval'

Name: Request approved
Template:
Type: Automatic action
Reference: A001004

☑ Execute actions immediately

Actions:

Name	Type
Validated request	Script

1 / 1

Request refused

The "Request refused" activity is similar to the "Request approved" activity.

In this case, the detail of the request needs to be modified in order to indicate that the request has been refused. The action to be executed is the "Request refused" action created in the preliminary step.

Creating the start event

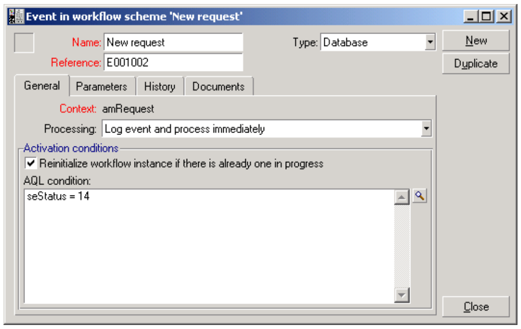
Events that trigger a workflow instance are associated with the "Start" activity.

To define a "Start" event, right-click the empty event zone in the "Start" activity and select **Detail of event**.

1. In our case, the workflow instance is triggered when the **Request status** field (SQL name: seApprStatus) 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:

   Figure 14.18. Activation parameters of start event

   Event in workflow scheme 'New request'

   Name: New request

   Reference: E001002

   Type: Database

   Context: amRequest

   Processing: Log event and process immediately

   Activation conditions

   ☑ Reinitialize workflow instance if there is already one in progress

   AQL condition:

   ```
   seStatus = 14
   ```

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:
   1. Check the **Update** box (SQL name: bUpdate).
   2. 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.

To create a transition:

1. Click the start event of the transition.
2. Hold the button down and drag to the target activity.

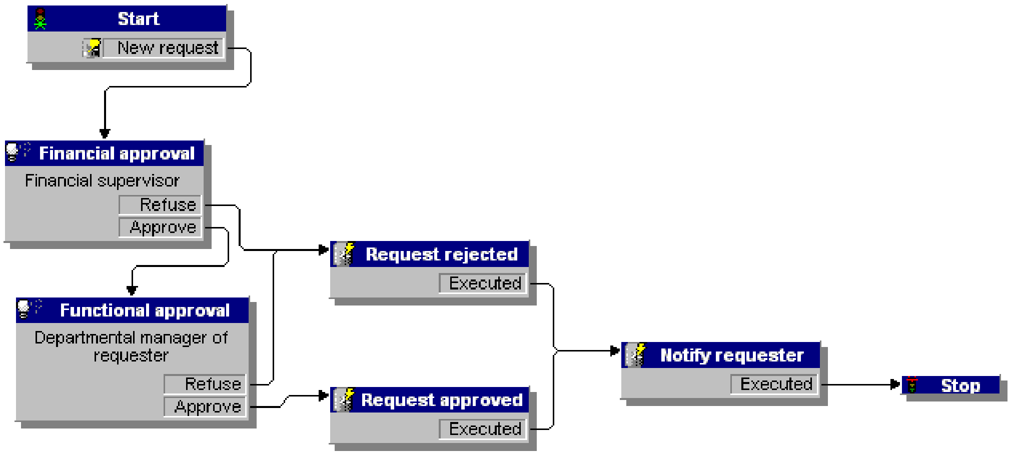
In this case, the transitions to be created are the following:

- 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:

Figure 14.19. Workflow 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:

1. Launch and configure AssetCenter Server.
2. Create the persons involved in the table of departments and employees.
3. Create a purchase request to be approved.
4. Trigger a workflow instance.
5. Control the instance.

Launching AssetCenter Server

1. Launch AssetCenter Server.

2 Connect to the AssetCenter database.

3 Select the **Action/ Activate** menu item. Then select "Search for new workflow execution groups" from the list of modules to be activated.

4 Click OK: AssetCenter Server verifies if any new workflow execution groups have been created and creates the specific monitoring modules for each of these groups.

> Note: Workflow execution groups classify workflow schemes. The groups to which a workflow scheme belongs is indicated in the Execution group field (SQL name: GroupName) in the General tab of the workflow detail.

In our example, we have not defined an execution group for the workflow scheme. AssetCenter Server thus creates a "Executing workflow rules for workflow schemes without execution group" monitoring module.

5 Leave AssetCenter Server running.

Creating records in the table of departments and employees

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 in order to perform the following operations:

1 Create the requester "Jerome Carpenter", belonging to the "IS department".

2 Attribute a **Login**, password and user profile allowing him to enter a purchase request (**Profile** tab in the detail of the corresponding record).

3 The supervisor of the "IS department" is "Philip Chavez".

4 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.

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 **Procurement/ Purchase requests** menu item under AssetCenter.
3. Click .
4. In the **Requester** field (SQL name: Requester) in the **General** tab of the detail of the request, select the record "Jerome Carpenter".
5. Set the **Request status** field (SQL name: seApprStatus) in the detail of the request to **In preparation**.
6. Confirm the purchase request: The start event occurs and an AssetCenter logs the event in the table of workflow elementary events (SQL name: "amWfOccurEvent").

Activating workflow

1. Launch the **Action/ Activate** menu item in AssetCenter Server. Select the **Execute workflow rules for workflow schemes without execution groups** module from the list.
2. Click OK: AssetCenter Server detects the event and activates the transition to the "Technical validation" activity corresponding to this event. In doing this, it creates a task to be carried out that is assigned to the requester's departmental supervisor.

Controlling the instance

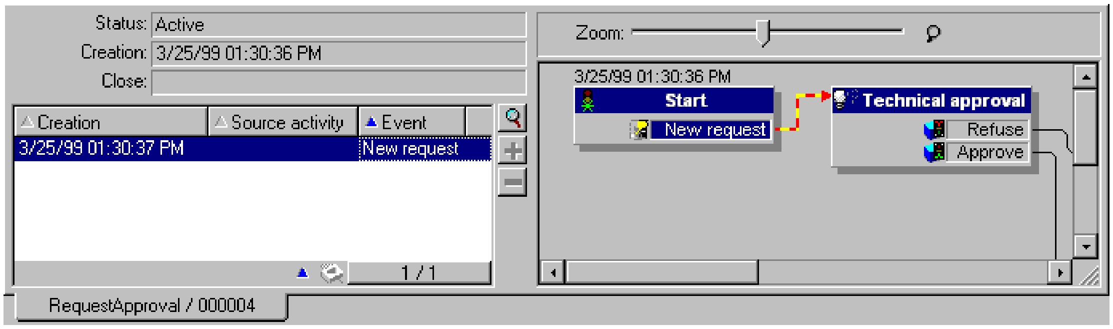
In order to verify that the workflow instance is functioning correctly, launch AssetCenter and connect to the demonstration database using the login of Philip Chavez, the departmental supervisor of "Jerome Carpenter".

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.

Figure 14.20. Current workflow instance for the purchase request



Status: Active
Creation: 3/25/99 01:30:36 PM
Close:

Creation	Source activity	Event
3/25/99 01:30:37 PM		New request

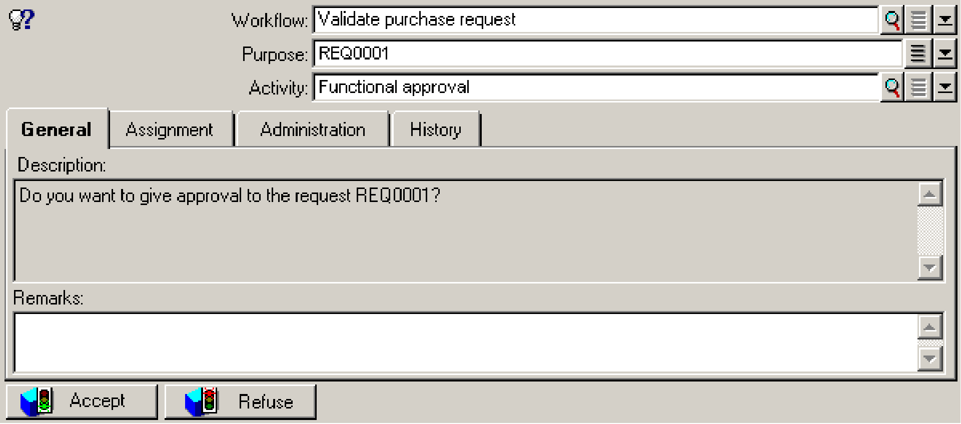
1 / 1

RequestApproval / 000004

Viewing the task to be carried out

1 Select the **Tools/ Tasks in progress** menu item: The departmental supervisor "Philip Chavez" can thus view the detail of the assigned task:

Figure 14.21. Detail of the task resulting from the functional approval activity



Workflow: Validate purchase request

Purpose: REQ0001

Activity: Functional approval

Description:

Do you want to give approval to the request REQ0001?

Remarks:

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 (i.e. the date of activation of the transition by AssetCenter Server).

You can click Detail to access the detail of the request giving rise to the task.

2 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.

The context of a workflow instance

Each workflow instance has its own specific context.

This section details the following points:

- Defining the context of a workflow instance.
- Object referenced by a workflow instance.

- Limiting workflow instances in progress for a given object.

Defining the context of a workflow instance

When defining a workflow scheme, you define:

- A default context at the top of the workflow detail.
- 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, i.e. 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)

AssetCenter 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:

Table 14.1. 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.

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 in order 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 table of workflow roles (SQL name: amWfOrgRole). Use the Tools/ Workflow/Roles menu item to access the table of workflow roles.

Workflow role type

The are several possible types of workflow roles (**Type** field (SQL name: seType) in the workflow role detail):

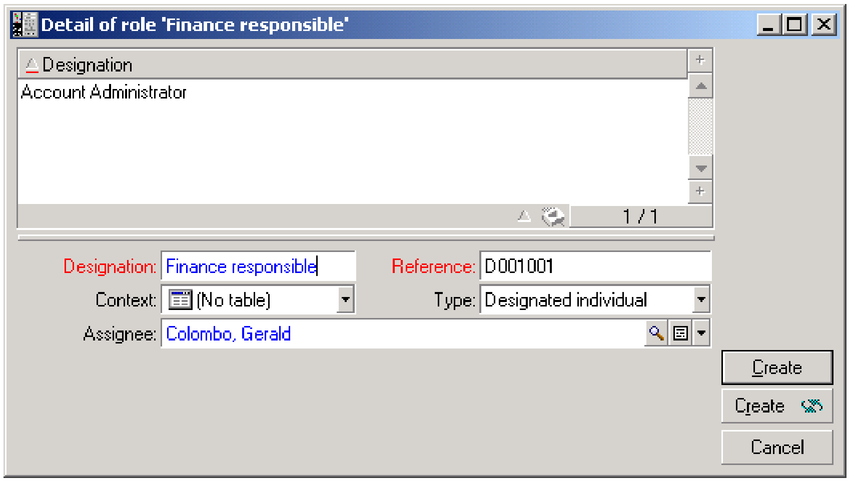
- **Designated individual.**
- **Calculated individual.**
- **Group.**

Designated individual

In this case, the assignee is selected in the table of departments and employees directly.

Example:

Figure 14.22. Example of a workflow role



Detail of role 'Finance responsible'

Designation
Account Administrator

1 / 1

Designation: Finance responsible

Reference: D001001

Context: (No table)

Type: Designated individual

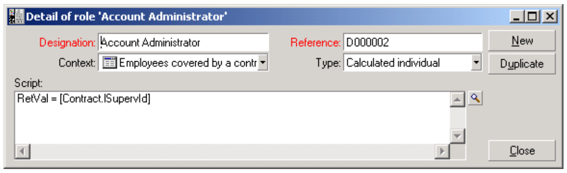
Assignee: Colombo, Gerald

Calculated individual

In this case, the assignee belongs to the table of departments and employees but is calculated by a script.

Example:

Figure 14.23. Example of a workflow role



Detail of role 'Account Administrator'

Designation: Account Administrator

Reference: D000002

Context: Employees covered by a contr

Type: Calculated individual

Script:

```
RetVal = [Contract.lSupervId]
```

Group

In this case, the assignee is selected in the table of employee groups (SQL name: "amEmplGroup").

Defining the assignee of an activity

The **Assigned** field (SQL name: Assignee) in the **Question** tab (**Question** type activity) or the **Action** tab (**User action** type activity) lets you define the assignee for a **Question** or **User action** type activity.

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:

- Question type activities.
- User-action type activity.
- Automatic-action type activity.
- Test/ Script type activities.
- Start activity
- Activity templates
- Triggering activities

Question type activities

These activities require the involvement of a user, specified in the **Assigned** 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.

Set the **Type** field (SQL name: seType) to **Question**. The **Question** tab will be displayed.

Specify:

1 The record in the table of workflow roles corresponding to the assignee. This assignee can be notified via an AssetCenter action. In order to do this, you just need to populate the **Notify person** field (SQL name: bNotifAssignee) in the **Question** tab appropriately.

Note: The action notifying the assignee is triggered as soon as the task to be performed is created; i.e. as soon as the transition triggering the activity is activated.

The assignee uses the **Tools/ Tasks in progress** 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, duplicate or delete a response, right-click the sub-tab label zone and select **Add linked record**, **Duplicate linked record** or **Delete link** from the shortcut menu.

Note: Each response automatically creates an output event for the activity.

User-action type activity

These activities require the involvement of a user, called the "assignee". The assignee is shown in the **Assigned** field (SQL name: Assignee).

Their definitions include:

- Instructions to follow.
- A wizard to execute.

Set the **Type** field to **User action** in order for the **Action** tab to be displayed.

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 AssetCenter action. In order to do this, you just need to populate the **Notify person** field (SQL name: bNotifAssignee) in the **Action** tab appropriately.

Note: The action notifying the assignee is triggered as soon as the task to be performed is created; i.e. as soon as the transition triggering the activity is activated.

The assignee uses the **Tools/ Tasks in progress** 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.

Automatic-action type activity

These activities are carried out automatically by AssetCenter or AssetCenter Server.

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.

Set the **Type** field to **Automatic action** in order for the **Actions** tab to be displayed.

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 AssetCenter Server or an agent in AssetCenter.

- 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 AssetCenter Server or an agent of AssetCenter executes the actions.
- Otherwise, the tasks are carried out by AssetCenter Server during its next verification cycle.

Test/ Script type activities

These activities are automatically performed by AssetCenter or AssetCenter Server.

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 products 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 product in stock.

Set the **Type** to **Test / script** in order for the **Test** tab to be displayed.

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, duplicate or delete a possible result, right-click the sub-tab label zone and select **Add linked record**, **Duplicate 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 AssetCenter Server or an agent in AssetCenter.

- 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 AssetCenter Server or an agent of AssetCenter executes the actions.
- Otherwise, the tasks are carried out by AssetCenter Server 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 Tools/ Workflow/Activity 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, etc.) to be automatically copied over 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 AssetCenter administrator.

Triggering activities

In order for an activity to be triggered, the **Input condition** field (SQL name: seInCond) 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 AssetCenter or AssetCenter Server) 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.

Tasks

This section explains how workflow tasks are created and executed:

- Creating tasks
- Automatic action or test/ script type activity
- Displaying the list of tasks in progress
- Performing a user task
- Assigning of user tasks
- Administrating a workflow task

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 **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.

Warning: 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 AssetCenter Server, or an AssetCenter agent.

Otherwise, AssetCenter Server verifies at regular intervals whether is needs to execute workflow task and executes them if necessary.

The frequency with which AssetCenter Server monitors workflow functions is defined in the options of AssetCenter Server.

Displaying the list of tasks in progress

The Tools/ Tasks in progress menu item allows you to display the list of tasks that need to be carried out.

The list displayed depends on the person connected to the database:

- An AssetCenter administrator can see all tasks in progress for all workflow instances.
- A workflow assignee sees:
    - The tasks they have to perform.
    - The tasks that are assigned to groups to which they belong but that are not assigned to a given person.

An administrator can also access the list of tasks in progress for a given activity from a workflow scheme detail. To do this:

1. Right-click an activity.
2. Select **Tasks in progress** from the shortcut menu.

Note: The displayed list is only a restricted view of the table of workflow tasks (SQL name: "amWfWorkItem": It shows the tasks that are to be carried out.

Performing a user task

A workflow assignee can access the list of tasks to be performed via the Tools/ Tasks in progress menu item.

Warning: If the user connected to the database is the AssetCenter administrator, the Tools/ Tasks in progress menu item displays all tasks to be carried out. Otherwise if the connected user is not the administrator, the Tools/ Tasks in progress menu item only displays those tasks that are assigned to them and those tasks assigned to groups to which they belong.

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.

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.

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.

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 describes workflow events and the way in which they are processed:

- System events
- Alarm events
- User events
- General conditions of activation
- Processing of events
- Application- Implementing a synchronous workflow scheme
- Terminal event

System events

System events are automatically defined by AssetCenter 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, called "Yes" and "No".

Alarm events

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 events

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.

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:
  - Check one or more appropriate boxes from among the **Insert** (SQL name: bInsert), **Update** (SQL name: bUpdate), and **Delete** (SQL name: bDelete) boxes.

    If you check the **Insert** box, records created are taken into account.

    If you check the **Update** box, you can specify the fields that, when modified, must be taken into account in the **Fields monitored** field (SQL name:MonitFields). To indicate several field names, use commas to separate them. If you leave the field empty, modified fields are not taken into account.

    If you check the **Delete** box, destroyed records are taken into account.

Warning: 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:

Figure 14.24. Parameters tab of a Database type action

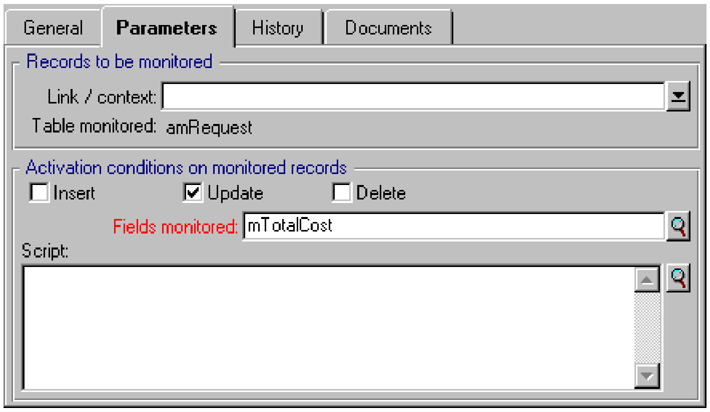


Table monitored: amRequest

Fields monitored: mTotalCost

As soon as a **Database** type event occurs, it is taken into account by the AssetCenter client machine on which it occurred. The way in which 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, please refer to the section entitled "Processing of events", in the "Workflow" chapter of this manual.

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 category "PC" are updated.

A **Periodical** type event occurs if the activation conditions indicated in the **General** tab are fulfilled.

In this case, AssetCenter Server triggers the event.

The frequency with which AssetCenter Server triggers **Periodical** type events is defined in the planner in the **Parameters** tab of the detail of the event.

The way in which 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:

Table 14.2. 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 in which 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:

Table 14.3. 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	AssetCenter Server 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, AssetCenter Server logs the event in the table with SQL name **amWfOccurEvent**. The transition is activated later by AssetCenter Server (the frequency with which AssetCenter Server monitors the transitions to be activated is defined in the options in AssetCenter Server).	AssetCenter Server 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, AssetCenter Server logs the event in the table with SQL name **amWfOccurEvent**. The transition is activated immediately by AssetCenter Server.	AssetCenter Server 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, AssetCenter Server does not log the event in the table with SQL name **amWfOccurEvent**. But the transition is activated immediately by AssetCenter Server.

	Log event and process by server	Log event and process immediately	Process event immediately without logging
Database type event or system event triggered by AssetCenter (result of a Question or User action type activity, result of a Automatic action or Test / script type activity executed by AssetCenter)	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by the AssetCenter client machine. The transition is activated later by AssetCenter Server (the frequency with which AssetCenter Server monitors the transitions to be activated is defined in the options in AssetCenter Server).	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by the AssetCenter client machine. The transition is activated immediately by the AssetCenter client machine.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by the AssetCenter client machine.
System event triggered by AssetCenter Server (result of a Automatic action or Test / script type activity executed by AssetCenter Server) or event on the activity alarm	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by AssetCenter Server The transition is activated later by AssetCenter Server (the frequency with which AssetCenter Server monitors the transitions to be activated is defined in the options in AssetCenter Server).	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by AssetCenter Server. The transition is activated immediately by AssetCenter Server.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by AssetCenter Server.

Table 14.4. The different ways in which an event can be processed

	Log event and process by server	Log event and process immediately
Periodical type event	AssetCenter Server 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, AssetCenter Server logs the event in the table with SQL name **amWfOccurEvent.** The transition is activated later by AssetCenter Server	AssetCenter Server 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, AssetCenter Server does not log the event in the table with SQL name **amWfOccurEvent**. But the transition is activated immediately by AssetCenter Server.
Database type event or system event triggered by AssetCenter (result of a Question or User action type activity, result of a Automatic action or Test / script type activity executed by AssetCenter)	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by the AssetCenter client machine. The transition is activated immediately by the AssetCenter client machine.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by the AssetCenter client machine.
System event triggered by AssetCenter Server (result of a Test / script or Automatic action type activity executed by AssetCenter Server) or event on the activity alarm	As soon as the event occurs, it is logged in the table with SQL name **amWfOccurEvent** by AssetCenter Server. The transition is activated immediately by AssetCenter Server.	When the event occurs, it is not logged in the table with SQL name **amWfOccurEvent** but the transition is activated immediately by AssetCenter Server.

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

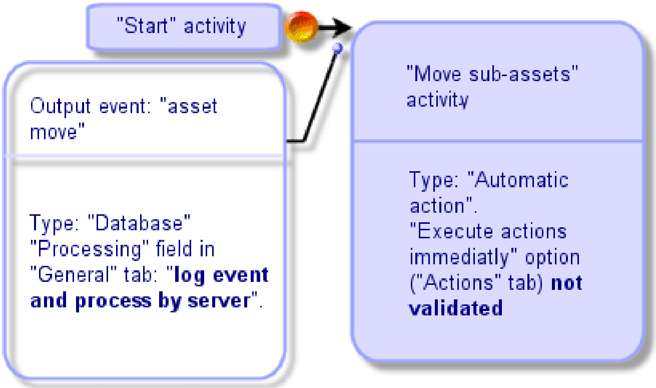
In order to accomplish a synchronous workflow scheme, you need to define:

- **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:

Figure 14.25. Example of synchronous workflow scheme


"Start" activity
Output event: "asset move"
Type: "Database" "Processing" field in "General" tab: "**log event and process by server**".
"Move sub-assets" activity
Type: "Automatic action". "Execute actions immediatly" option ("Actions" tab) **not validated**

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.

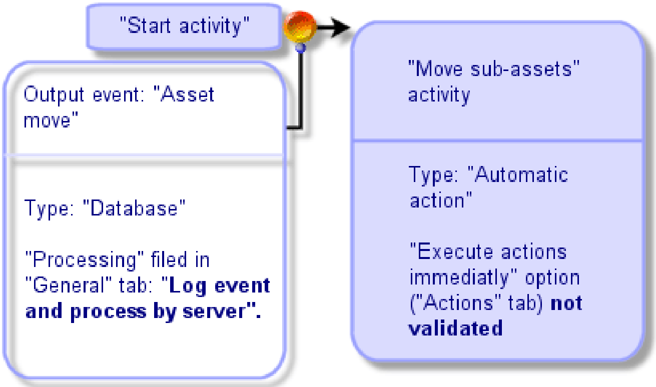
3 The workflow instance starts up.
4 The workflow transition is activated.
5 The location of the sub-assets is modified.
6 Then the full transaction is validated.

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.

Figure 14.26. Example of asynchronous workflow scheme


"Start activity"
Output event: "Asset move"
Type: "Database"
"Processing" filed in "General" tab: **"Log event and process by server".**
"Move sub-assets" activity
Type: "Automatic action"
"Execute actions immediatly" option ("Actions" tab) **not validated**

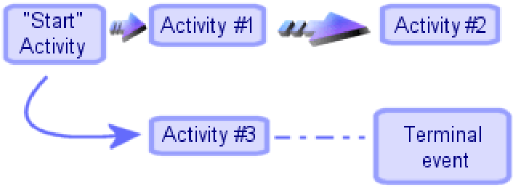
Terminal event

Definition

A terminal event ends a workflow instance, even if there are remaining tasks to be performed.

Example:

Figure 14.27. 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.

Note: Warning: Without the Log task option (SQL name: bLogWorkItem) in the General tab of the activity detail validated, it is not possible to define time-limits or alarms.

This section details the following points:

- Time limit
- Workflow alarms

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**

Note: Warning: If you specify a period of time, AssetCenter 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 AssetCenter Server. The frequency of monitoring is defined in the options of AssetCenter Server.

The effect of alarms

Alarms trigger:

- AssetCenter 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 classify 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.

AssetCenter Server monitors the creation of new workflow execution groups.

As soon as AssetCenter Server 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 AssetCenter Server.

Once a workflow execution group is detected, AssetCenter Server monitors and executes the corresponding workflow rules (monitoring alarms, processing of **Periodical** type events, activating transitions, executing tasks, etc).

Workflow tracking

When a table in AssetCenter 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.

Each workflow instance is described in a sub-tab that describes the progress of the instance:

- The left part of the sub-tab list the events that have occurred.
- The right part of the sub-tab displays the workflow scheme. The activities to be carried out are shown as flashing. The following steps are grayed out.

15 Importing data

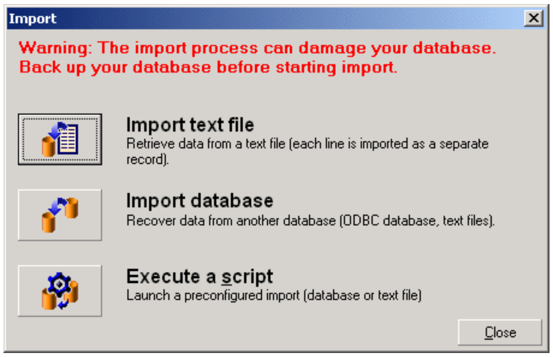
CHAPTER

This chapter explains how to import data with AssetCenter.

To import a single text file use the **File/ Import** menu, then the **Import text file** option.

To import several text files or a database use the File/ Import menu, Import database option.

Figure 15.1. Selecting the type of data to be imported



Import

Warning: The import process can damage your database.
Back up your database before starting import.

Import text file
Retrieve data from a text file (each line is imported as a separate record).

Import database
Recover data from another database (ODBC database, text files).

Execute a script
Launch a preconfigured import (database or text file)

Overview of importing data

An AssetCenter administrator can import data into the AssetCenter database from:

A single text file

The text file is mapped to a main table in the AssetCenter database.

Each field in the text file is mapped to a database field, which can be located in the main table or in a linked table.

Several text files

Each text file is mapped to a table in the AssetCenter database.

Each field in each text file is mapped to a database field, which can be located in the main table or in a linked table.

A complete database

You can import a database:

- AssetCenter
- ODBC for other databases

Each table in the source database is mapped to a table in the target database.

Each field in the source table is mapped to a field in a table of the target database. The target field can be located in the table that is mapped directly or in a linked table.

You can add or modify AssetCenter records. You cannot delete AssetCenter records.

All the information concerning the transfer can be saved as a script. You can use the script to import the data again without re-defining import settings.

The import module gives you the choice of several ways to handle errors and the possibility to write the results of operations carried out to a log file.

Note: Protection of databases: Only an administrator of AssetCenter can use the File/ Import menu item ("Admin" login or user with administration rights). This menu item is disabled for other users, which protects access to the database.

Note: Importing numeric data: Numeric fields must be formatted appropriately, regardless of the options defined in Windows Control Panel: Only numeric characters without spaces are allowed, and the period "." is used as decimal separator. Numeric fields must be imported into numeric fields. This way, numeric data can be imported independently of the settings of the computer or computers used to perform the import.

Note: Importing images, forms, queries, reports, floor plans, access rights, views, passwords and "logins": You can only import these items by importing an AssetCenter database.

Recommendations

Here are some recommendations before importing data into the AssetCenter database:

Default values for mandatory fields

The import module takes into account the mandatory nature of fields. If a record to be imported contains an empty field, and if this field is defined as mandatory in the target database, the import module rejects this record. To avoid records being rejected for this reason, we recommend assigning default values to mandatory fields in the target database. If a value is given in the files to be imported this overrides the default value of the field.

Avoid using "Id" field as reconciliation keys

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.

Backup the AssetCenter database

Since importing can make global modifications to your AssetCenter database, we recommend performing a backup before launching the import.

Avoid simultaneous access to the AssetCenter database

We advise against performing simultaneous imports on different machines or using AssetCenter on another machine during an import.

Constraints to be respected when importing data into a field

When importing data into the AssetCenter database, the import module verifies that the data is compatible with the structure of the database. If certain fields or links of data items to be imported are incompatible with the structure of the database, the outcome will be one of the following cases:

- The record is rejected as a whole.
- The value is truncated.

- The field is left as is.

It is therefore preferable to make sure that data presented for import is in phase with the structure of the database in order to avoid rejections.

Two possibilities are worth considering:

- You are importing a structured database:
  - If the source and target fields are of the same type (source-date and target-date for example), or if the fields are compatible (source-date+time and target-date for example), there are no particular constraints to be respected.
  - If the source field is a "text" type field and the target field is of a more specific type, you must respect the constraints of the target field.
- You are importing data from a text file:
  - You must respect the constraints of the target field.

According to the value of the "UserType" property of the target field

Table 15.1. Constraints to be respected when importing data into a field according to the value of the "UserType" property of the target field

If the value is:	Then:
Default	You must respect the format defined by the "Type" property.
Number or Money	The source field must be a number. If the source field originates from a structured database and the field type is "number" or "money" then there is no constraint. If the source field is a "text" type field, you must present a number, use the period "." as a decimal separator an do not use a separator for thousands.
Yes/No	The source field must show "1" (for "Yes"), or "0" (for "No").
Date	If the source field originates from a structured database and the field type is "Date" or "Date+Time", then there is no constraint. If the source field is a "text" type field, you must respect the following constraints: • The date format (order of year, month and day) must be the same for all records. Specify this format when importing. • You must systematically enter a day, month and year. • Use the same separator (of your choice) to separate days, months and years for all records. Specify this separator when importing.
System itemized list	Example If the itemized list is "Yes|1|No|0", importing "Yes" or "1" will give the same results. You must present one of the values of the itemized list only. Otherwise, the line will not be imported. You can identify the item from the itemized list by its clear value or as its number as stored in the database. If you present an empty value, the import module attributes the value "0" to the field. We recommend presenting the stored numeric values, since these are more stable than their text equivalents from one version of AssetCenter to another and independent of the language version used.
Custom itemized list	The value of the field is selected in an itemized list, which can be modified by the user. You can therefore present the import module with one of the values of the itemized list. If, in addition, the itemized list is "open", you can present any value. This value will be added to the itemized list.
Percentage	Imported values must be the percentage value with or without the percentage sign "%" (for example: "10" or "10%").
Time span	You must respect the constraints defined by the "UserType" and "UserTypeFormat" properties with UserType is "Time span".

If the value is:	Then:
Table or field SQL name	You can import any alpha-numeric value. But if this value does not correspond to a valid SQL name of a field or table, you run the risk of corrupting the database.

According to the value of the "type" property of the target field

You must respect the constraints linked to this property if the "UserType" property is set to "Default".

Particular case:

Table 15.2. Constraints to be respected when importing data into a field according to the value of the "Type" property of the target field

Special case: If the value is:	Then:
Date+Time	If the source field originates from a structured database and the field type is "Date" or "Date+Time", then there is no constraint. If the source field is a "text" type field, you must respect the constraints defined by the "UserType" property when the UserType property is "Date" or "Date+Time".

According to the value of other properties of the target field

Table 15.3. Constraints to be respected when importing data into a field according to the value of other properties of the target field

If this property:	Is set to:	Then:
MandatoryType	Yes	If the source field is empty, the records that the import module should have added or modified are left untouched.
Size	Is populated	Values from the source field that are too long are truncated on import.
ReadOnly	Yes	It is not possible to import a value to a field with this property.

"dtLastModif" field (SQL name)

This field is updated when you modify or create records in the AssetCenter database via the user interface or via an import. It indicates the date of modification or creation of the record.

If you import a value to this field, this will overwrite the real import date.

Importing text files or an ODBC database

This section explains how to import one or more data files or an ODBC database.

To import a single text file use the File/ Import menu item, Import text file option.

To import a set of text files, use the File/ Import menu item, Import database option, Text tab.

To import an ODBC database use the File/ Import menu item, Import database option, ODBC tab

Before importing text files

Start by preparing the files containing the data to be imported. They must meet the following conditions:

- Each column represents a field.
- Each line represents a record.
- You may optionally include the field names at the start of the file. If you do not include them in the file, you may define them when you import the data.
- At the start of the file, before the optional line containing the field names, you can include up to 99 lines of comments that will not be imported: This can be done by populating the "First import line" field. These comment lines do not have to start with a special character.
- You may create the file using the OEM (DOS), UTF-8, UNICODE or Latin 1 character set.
- Columns can have either a fixed width or a variable width. In this case you select a character to act as a separator.
- Field contents may be delimited by any characters you choose.

- Field types may be numeric, character, or date.
- Imported values must follow the constraints related to the AssetCenter database structure (data entry format, field type, link type, index type.)
- All characters are allowed except for those you use to delimit the text. There is no way to include the delimiter character in a text string.
- Date, date+time and duration data are subject to the same constraints as if they were directly entered into the program.
- We recommend that you build one text file for each main target table.

If your source data resides in a database whose engine is not supported by the import module, you must extract the data to text files and import them afterwards. If the database is recognized by AssetCenter you can import the information directly from the database using the **File/ Import** menu item, **Import database** option.

Step 1: Select the text files or ODBC database to be imported

Importing a single text file

1. Select the **File/ Import** menu item.
2. Select the **Import text file** option.
3. AssetCenter asks you to indicate the location of the structured text file containing the data to be imported.

Importing a set of text files

To select the text files you have prepared:

1. Select the **File/ Import** menu item, **Import database** option, **Text** tab.
2. Click Open....
3. Once in the import module, select the **File/ Add file** menu item. Indicate the names of the text files to be added.

Importing an ODBC database

Use the **File/ Import** menu item and choose the option marked **Import text file** to select the source database.

Using the **ODBC** tab you can import an ODBC database.

1. Identify the data source, the user and the password. The ▣ icon lets you directly create an ODBC data source, without having to go through the Windows Control Panel then the ODBC Data Source Administrator.
2. Click Open....

Step 2: Define the parsing of files or tables to be imported

Important: When the file contains fixed width fields, this step involves only one window. Two windows are necessary when the file contains separators between fields.

When importing a single text file

Once you have selected the text file in step 1, AssetCenter automatically asks you how it should be parsed.

When importing a set of text files

Once you have selected the text file in step 1, AssetCenter automatically asks you how it should be parsed. You may perform this step later by clicking OK, or fill in this information at once.

There are different ways to recall this window when you are in the main screen of the import module:

- Double-click the source file.
- Or use the **Edit/ Properties** menu item after having selected the source file.

The list of text files can be found in the "Source tables" column in the main screen of the import module.

When importing an ODBC database

Once you have opened the database in step 1, you can display the description of each table from the import module's main screen:

- Double-click the source table.
- Or use the **Edit/ Properties** menu item after having selected the source table.

The list of source tables can be found in the "Source tables" column in the main screen of the import module.

First screen

Character coding

Indicate if the text uses the ANSI, OEM(DOS), UTF-8, UNICODE or Latin 1 character set.

First import line

Enter the line number containing the first line of data to be imported. Lines before this number are skipped by the import module.

If your document contains a line with field names and if this line is located just before the first data line, enter the number of the line with field names.

AssetCenter can skip up to 99 lines at the start of the file.

First imported line contains column names

Select this check box if the first line to be imported contains the field names (the column names). This way you do not have to enter the column names yourself.

If your file does not contain column names, you can define them in the following steps.

By separators

Check this check box if the field values are separated by a particular character.

Specify this character in the following screen.

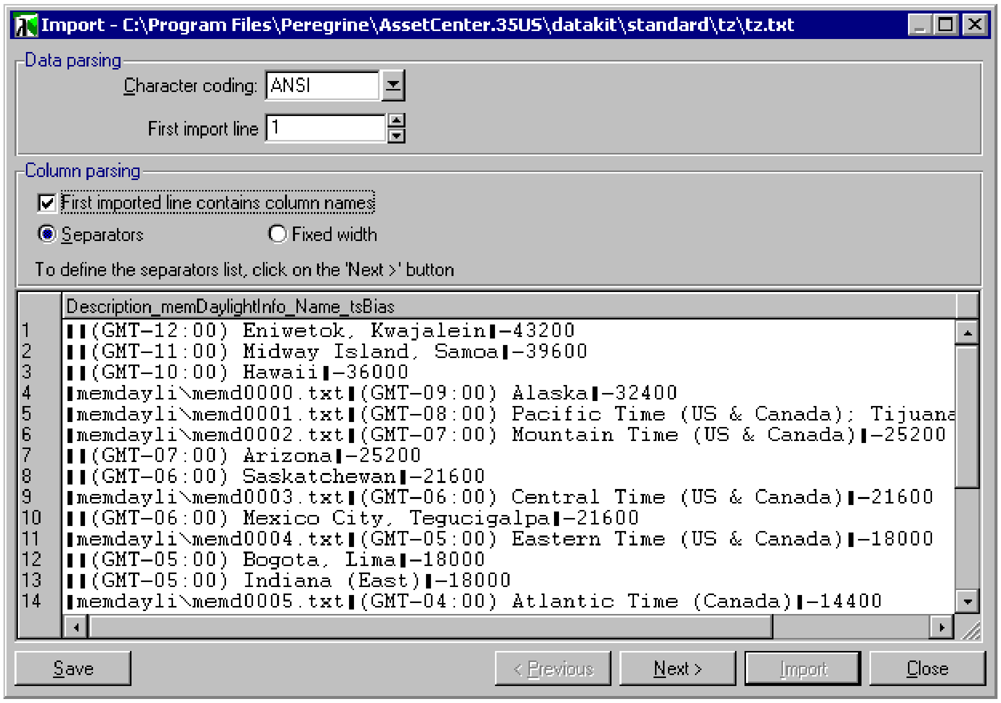
Fixed width

Check this check box if all the values of each field are the same length.

AssetCenter automatically displays column boundaries.

- To move a column boundary, use the mouse to select the boundary in the data area and drag it to the desired position. You cannot select the boundary in the title area.
- To remove a column boundary, use the mouse to select the boundary and drag it outside the table.
- To add a new column boundary, click with the mouse in the data area at the position where the separator should be inserted.

Figure 15.2. Defining how to parse text files (screen 1)



Import - C:\Program Files\Peregrine\AssetCenter.35US\datakit\standard\tz\tz.txt

Data parsing

Character coding: ANSI

First import line 1

Column parsing

☑ First imported line contains column names

● Separators ○ Fixed width

To define the separators list, click on the 'Next >' button

```
   Description_memDaylightInfo_Name_tsBias
1  ▌▌(GMT-12:00) Eniwetok, Kwajalein▌-43200
2  ▌▌(GMT-11:00) Midway Island, Samoa▌-39600
3  ▌▌(GMT-10:00) Hawaii▌-36000
4  ▌memdayli\memd0000.txt▌(GMT-09:00) Alaska▌-32400
5  ▌memdayli\memd0001.txt▌(GMT-08:00) Pacific Time (US & Canada); Tijuana
6  ▌memdayli\memd0002.txt▌(GMT-07:00) Mountain Time (US & Canada)▌-25200
7  ▌▌(GMT-07:00) Arizona▌-25200
8  ▌▌(GMT-06:00) Saskatchewan▌-21600
9  ▌memdayli\memd0003.txt▌(GMT-06:00) Central Time (US & Canada)▌-21600
10 ▌▌(GMT-06:00) Mexico City, Tegucigalpa▌-21600
11 ▌memdayli\memd0004.txt▌(GMT-05:00) Eastern Time (US & Canada)▌-18000
12 ▌▌(GMT-05:00) Bogota, Lima▌-18000
13 ▌▌(GMT-05:00) Indiana (East)▌-18000
14 ▌memdayli\memd0005.txt▌(GMT-04:00) Atlantic Time (Canada)▌-14400
```

Note: The frame containing the records displays a partial preview of the file to be imported. A maximum of 25 lines are shown.

Possible second screen

This second screen is shown if in the first screen:

1 You specify that the field values are separated by a separator characters.

2 You click Next >.

Column separators

Indicate the character used to separate two successive field values. We recommend using the semi-colon character " ;" as separator.

If you check the **Treat consecutive delimiters as one** option, AssetCenter will treat identical consecutive delimiters as one and will not create empty columns. If you want to created an empty column, uncheck this box use two consecutive delimiters in your text file.

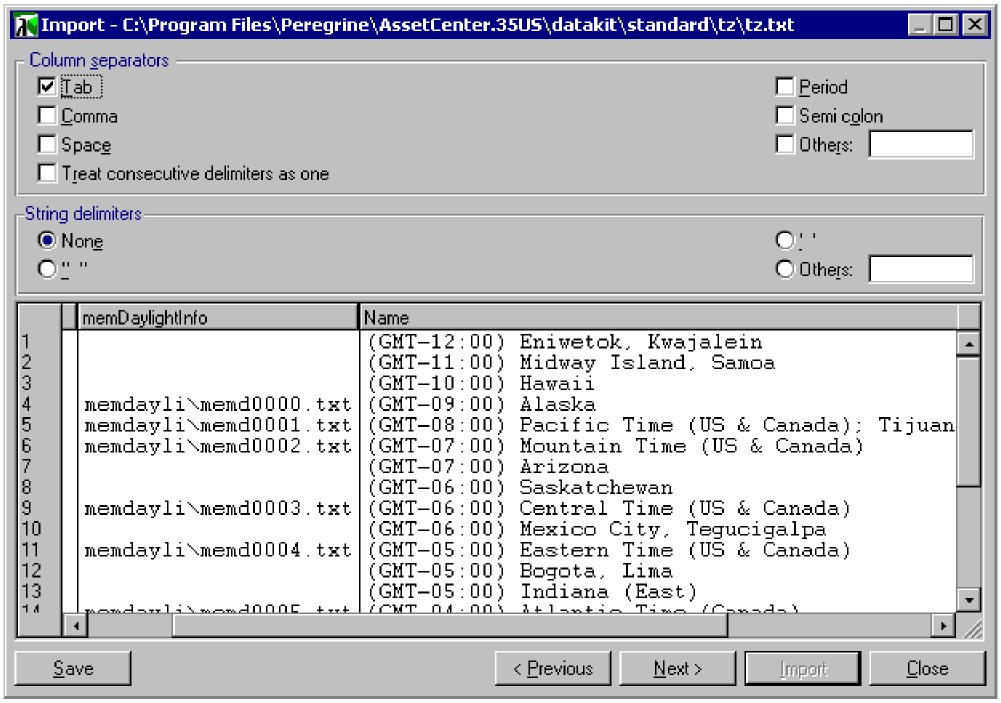
String delimiters

If you use character to delimit text, indicate the character that you use. AssetCenter will remove these extra characters if they are present, before transferring the field to the database.

If, between two delimiters, AssetCenter encounters a column separator, it is considered as text. Specifying a string delimiter does not oblige to you use it systematically around all values. On the other hand, if you place a string delimiter before the start of a string this must be balanced by a delimiter at the end of the string.

You cannot import a string delimiter as a value.

Figure 15.3. Defining how to parse text files (screen 2)



Import - C:\Program Files\Peregrine\AssetCenter.35US\datakit\standard\tz\tz.txt

Column separators

☑ Tab
☐ Comma
☐ Space
☐ Treat consecutive delimiters as one
☐ Period
☐ Semi colon
☐ Others:

String delimiters

● None
○ " "
○ ' '
○ Others:

	memDaylightInfo	Name
1		(GMT-12:00) Eniwetok, Kwajalein
2		(GMT-11:00) Midway Island, Samoa
3		(GMT-10:00) Hawaii
4	memdayli\memd0000.txt	(GMT-09:00) Alaska
5	memdayli\memd0001.txt	(GMT-08:00) Pacific Time (US & Canada); Tijuan
6	memdayli\memd0002.txt	(GMT-07:00) Mountain Time (US & Canada)
7		(GMT-07:00) Arizona
8		(GMT-06:00) Saskatchewan
9	memdayli\memd0003.txt	(GMT-06:00) Central Time (US & Canada)
10		(GMT-06:00) Mexico City, Tegucigalpa
11	memdayli\memd0004.txt	(GMT-05:00) Eastern Time (US & Canada)
12		(GMT-05:00) Bogota, Lima
13		(GMT-05:00) Indiana (East)
14	memdayli\memd0005.txt	(GMT-04:00) Atlantic Time (Canada)

Step 3: Describe the fields to be imported

When importing a single text file

Once you have parsed the text file as explained in step 2, click the Next > button to display the screen for describing the fields in the file.

When importing a set of text files

Once you have parsed the text file as explained in step 2, click the Next > button to display the screen for describing the fields in the file.

You can display this screen from the import module's main window by double-clicking the source file or by using the **Edit/ Properties** menu item after selecting the file and clicking the Next > button once or twice, according to the "column parsing" selected in the previous step.

When importing an ODBC database

Once you have selected the database in step 1, you can display the description of the tables from the import module's main screen. Either double-click the source table or use the **Edit/ Properties** menu item after having selected the source table (the "Source tables" column in the main screen of the import module displays the list of source tables.)

Select the column to be configured by clicking anywhere in the column.

Number

The number of the selected column is displayed here.

If you did not select the **First line contains column names** field in step 2, you can select the column number directly, instead of clicking the table.

Name

The column name (or field name) appears here.

1. If you selected the **First line contains column names** field in step 2, you cannot modify the column name.
2. If you did not select that option, either leave the default name or modify it. This name makes it easy to identify columns during later steps.

Type

Enter the field type to be imported. AssetCenter lets you choose from the following possibilities:

- Number: All characters must be numbers. If AssetCenter finds other types of characters, this field 's value is set to "0".
- Character string: All characters are allowed except those you used as text delimiters.

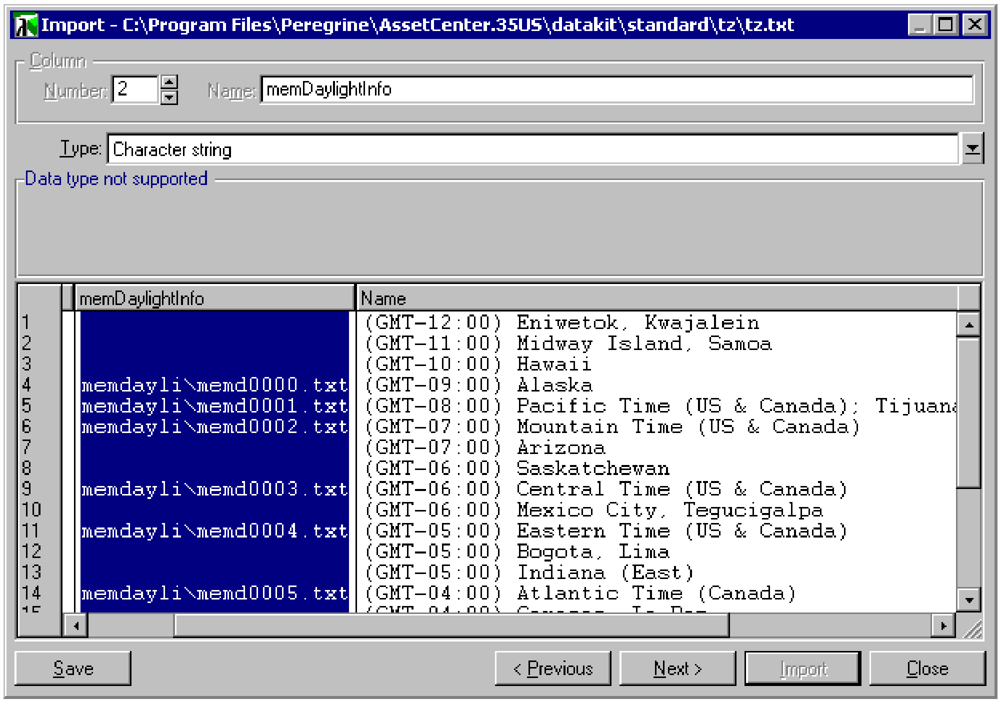
- Date: Only the date formats defined in the "Date format" frame are accepted. If AssetCenter finds other formats when importing, the field's value is set to null.

Date format

If you indicate that the field contains a "Date", AssetCenter asks you for the separator character between the day, month and year, as well as the order in which they appear.

Besides these two parameters, your dates may use all the possibilities available for entering dates in AssetCenter.

Figure 15.4. Describing fields in a source text file



Import - C:\Program Files\Peregrine\AssetCenter.35US\datakit\standard\tz\tz.txt

Column

Number: 2 Name: memDaylightInfo

Type: Character string

Data type not supported

	memDaylightInfo	Name
1		(GMT-12:00) Eniwetok, Kwajalein
2		(GMT-11:00) Midway Island, Samoa
3		(GMT-10:00) Hawaii
4	memdayli\memd0000.txt	(GMT-09:00) Alaska
5	memdayli\memd0001.txt	(GMT-08:00) Pacific Time (US & Canada); Tijuana
6	memdayli\memd0002.txt	(GMT-07:00) Mountain Time (US & Canada)
7		(GMT-07:00) Arizona
8		(GMT-06:00) Saskatchewan
9	memdayli\memd0003.txt	(GMT-06:00) Central Time (US & Canada)
10		(GMT-06:00) Mexico City, Tegucigalpa
11	memdayli\memd0004.txt	(GMT-05:00) Eastern Time (US & Canada)
12		(GMT-05:00) Bogota, Lima
13		(GMT-05:00) Indiana (East)
14	memdayli\memd0005.txt	(GMT-04:00) Atlantic Time (Canada)

Note: The table shows up to 25 lines of the file to be imported.

Step 4: Map source fields to target fields in the AssetCenter database

Importing a single text file

Once you have selected the source fields as described in step 3, click the Next > button to display the screen for mapping fields in the text file to AssetCenter database fields.

1 Start by mapping the text file to a table in the AssetCenter database using the "Target table" field.
2 Then map the fields in the text file to be imported (displayed in the "Target table" table) to a field in the AssetCenter database. (The fields in the target table and its linked tables are displayed in the list on the right under the "Target table" field.)

Importing a set of text files

1 Once you have selected the source fields as described in step 3, click the OK button to return to the import module's main window.
2 Map each text file to a target table.
3 Next, for each (text file, target table) pair in the table on the right, map each text field to an AssetCenter table: Either double-click the pair, or select it and use the **Edit/ Properties** menu item.

Importing an ODBC database

1 Once you have selected the source fields as described in step 3, click the OK button to return to the import module's main window.
2 Map each text file to a target table.
3 Next, for each (text file, target table) pair in the table on the right, map the text fields to an AssetCenter database field: Either double-click the pair, or select it and use the **Edit/ Properties** menu item.

Step 5: Map each text file or source table to a target table

Note: This section is relevant when importing a set of text files or an ODBC database.

Map each text file or source table to a target table

Click the file or the source table ("Source tables" column) and on the corresponding target table (in the "Target tables" column). Then do one of the following:

- Either, use the **Edit/ Map** menu item/
- Or, click the icon.
- Or, use the **Edit/ Map** by name menu item: AssetCenter automatically maps the files or tables that have exactly the same name. This is achieved using the technical name of the field.

Use the **Edit/ Unmap** menu item or the icon to unmap a file or a source table from a target table.

Display the AssetCenter database structure

Double-click the target table in the "Target tables" column, or select it with the mouse and use the **Edit/ Properties** menu item. AssetCenter displays the list of fields, their type and length.

Step 6: Map the fields to be imported to fields in the AssetCenter database

Source fields

This part of the screen displays the names you assigned to columns in step 3 (for text files) or the short field description (for AssetCenter database.)

Target table

When importing a single text file

Select the target table that will receive the data. AssetCenter displays the table's structure (fields in the table or in linked tables.)

When importing a set of text files or an ODBC database

AssetCenter displays the structure of the target table mapped to the file or the source table (fields in the table or in linked tables).

Map source fields to target fields

There are several possibilities available to you:

- Use the mouse to drag a "source field" to a "target field" in order to map them.
- You can also select a "source field", select a "target field" then click the icon to map them.
- The icon lets you unmap a source field from a target field after having clicked the (source field, target field) pair.
- The icon allows you to automatically map the source fields and target fields that share the exact same name. This is achieved using the technical name of the field.

Add additional calculated fields in the source file

AssetCenter allows you to add additional fields to your source file. These fields are not saved; just placed in memory.

Use the , and icons to add, delete or display these additional fields.

Selecting keys

You may select one or more target fields to create record identification keys. An identifier key lets you identify a record in a table. If you select several keys, all these keys will let you identify the records.

Select the (source field, target field) pairs to qualify and click the button to declare them as "keys". If this button is active, it appears as if it were pressed in and is lighter; the small icon to the left of the target field or link also looks like 🟥.

AssetCenter imports lines from the source file one by one, proceeding as follows:

- If there is a record in the database whose keys have exactly the same values, AssetCenter modifies the record according to the information contained in the text file.
- If several records exist with the same set of keys, the program stops at the first record it finds and ignores the others. It is therefore up to you to select the appropriate keys.
- If no records exist that match the keys, AssetCenter creates a new record in the database.

Note: We advise against using the "Id" fields of tables as reconciliation keys if you want to reimport data you have exported. Indeed, 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.

Configuring the creation of linked records

When you import a file containing data to be imported into several different tables (a file containing employees and their locations, for example), select a main target table (the employee table in our example) and use links to indicate where the data should be imported into other tables (the locations table in our example.)

AssetCenter lets you configure how records will be created in the linked table, if the record does not exist before the import process. Use the icon to perform this configuration. This icon can only be used on links (not on the fields of linked tables.) Links are represented by the and icons.

To display the configuration screen:

1. Map the field to be imported to the field in the linked table.
2. Click the corresponding link.
3. Click the icon.

If the record is found using the specified identification keys, the program modifies the information in the record, if appropriate.

Create the record

The record is created if it was not found in the database using the specified identification keys.

Do not create the record

Records are not created, even if they are not found in the database using the specified identification keys.

Signal error - abnormal situation

AssetCenter generates an error message if the record is not found in the database using the specified identification keys.

Take into account only records that are already linked (📌)

The import module only considers records that are already linked to the main record if you attach the 📌 icon (pushpin) to the link.

Example

Example: You are importing a list of employees and the assets they use. The table of departments and employees is the target table. You attach a pushpin to the link with table of assets. For each asset associated with an employee in the source file, the import module only takes into account the assets that they are already using (**Assets used** tab in employee detail).

In the case of "Own" type links, the pushpin is automatically placed and cannot be removed. "Own" type links are links for which the linked records are automatically deleted if the main record is deleted. The employees-training link is an example of this type of link: If you delete an employee, you will also delete all the training item linked to them.

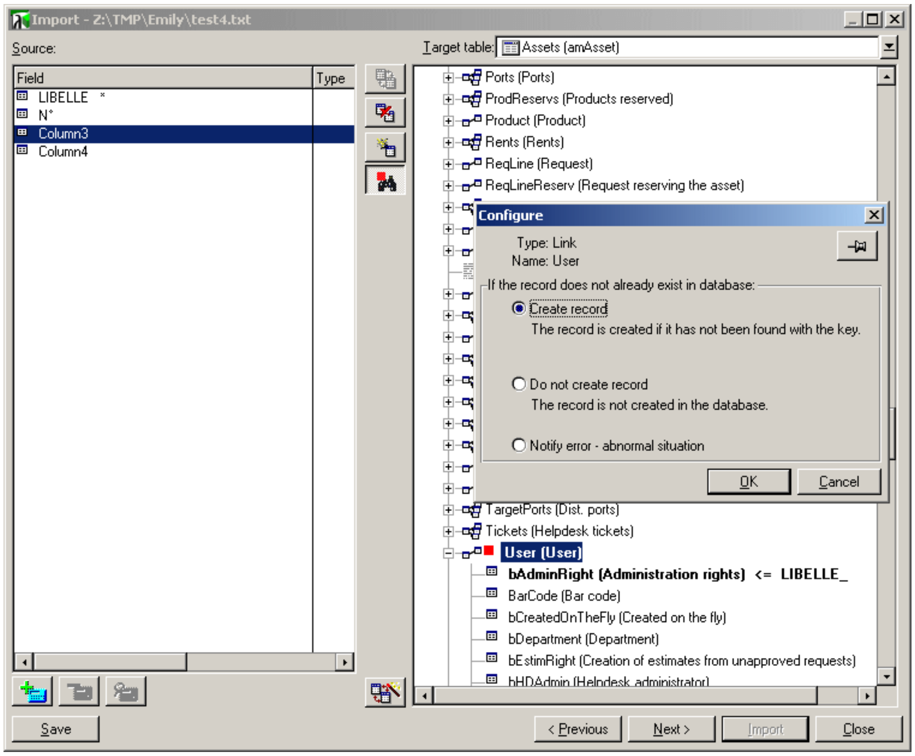
The effect of the pushpin depends on the types of links:

- If the target table is the table of assets, and you attach a pushpin to the "user" link, the import module only looks for users linked to the assets. Since there is only one user for a given asset, it is thus possible to modify or create the user of an asset without having to identify this user by a key. This is also very useful for modifying the value of a feature of a given asset.

- If the target table is the table of departments and employees, and you attach a pushpin to the "assets" link, the import module only looks for the assets linked to the employee. In this case, to modify or create the assets of a user, you do have to have the appropriate identification keys, but they do not have to be as specific as they would be when not using the pushpin.

Note: AssetCenter uses three kinds of links between records: n links: For example, an asset can only be linked to one location, but a location can be linked to several assets. 1 links: For example, an asset can only be linked to one remark and a remark can only be linked to one asset. n-n links: For example, a supplier can be linked to several products; a product can be linked to several suppliers.

Figure 15.5. Configuring how records are created in linked tables



Import - Z:\TMP\Emily\test4.txt

Source:

Field	Type
LIBELLE *	
N°	
Column3	
Column4	

Target table: Assets (amAsset)

- Ports (Ports)
- ProdReservs (Products reserved)
- Product (Product)
- Rents (Rents)
- ReqLine (Request)
- ReqLineReserv (Request reserving the asset)
- TargetPorts (Dist. ports)
- Tickets (Helpdesk tickets)
- User (User)
  - bAdminRight (Administration rights) <= LIBELLE_
  - BarCode (Bar code)
  - bCreatedOnTheFly (Created on the fly)
  - bDepartment (Department)
  - bEstimRight (Creation of estimates from unapproved requests)
  - bHDAdmin (Helpdesk administrator)

Configure

Type: Link
Name: User

If the record does not already exist in database:

● Create record
The record is created if it has not been found with the key.

○ Do not create record
The record is not created in the database.

○ Notify error - abnormal situation

Symbols used in the tree structure of target tables

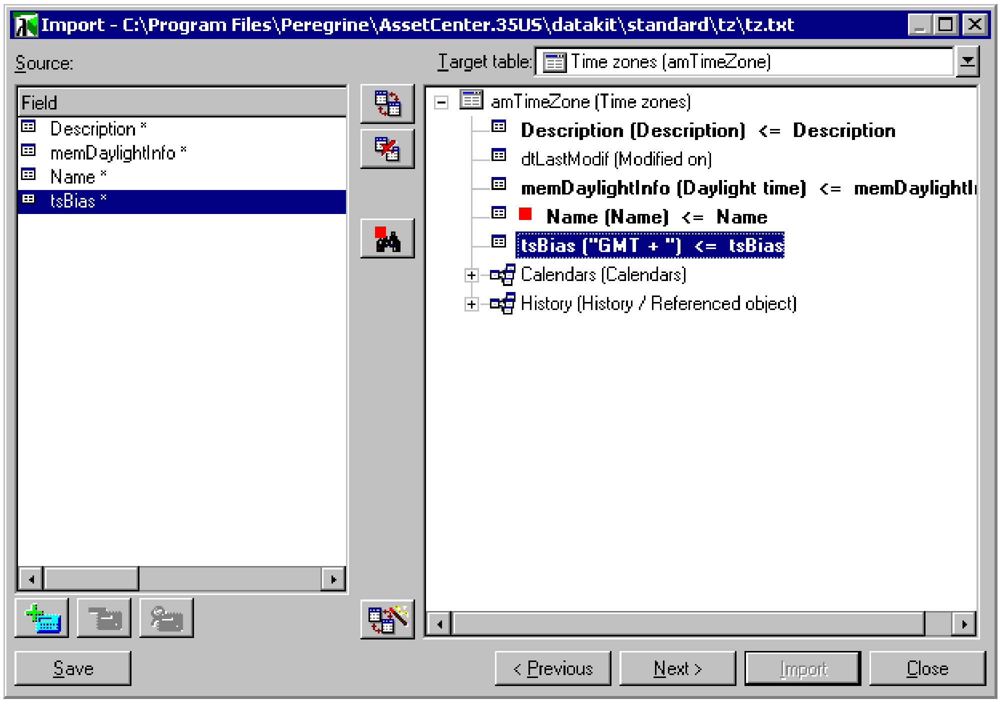
Indicates that the name following the symbol is a table (large symbol) or a field (small symbol).

Indicates that the name following the symbol is a table linked to its parent table. You can only choose one record in the linked table from a record detail in the parent table. This type of link represents a field whose value can be entered using a "selection window" or a "drop down list".

The name following the symbol is a table linked to its parent. You can choose several records in the linked table from a record detail in the parent table. This type of link represents a list of records located in a tab of the detail window of the parent table.

📌 This symbol is called the "pushpin". It can only be attached to a link to a table containing a field to which the field to be imported is mapped. When a pushpin is set, the import module only looks among records linked to the record to which it is "pinned". The presence of the pushpin is determined by the options displayed using the ▣ icon.

Figure 15.6. Mapping source fields to target database fields



Import - C:\Program Files\Peregrine\AssetCenter.35US\datakit\standard\tz\tz.txt

Source:

Field
- Description *
- memDaylightInfo *
- Name *
- tsBias *

Target table: Time zones (amTimeZone)

- amTimeZone (Time zones)
  - **Description (Description) <= Description**
  - dtLastModif (Modified on)
  - **memDaylightInfo (Daylight time) <= memDaylightI**
  - 🟥 **Name (Name) <= Name**
  - **tsBias ("GMT + ") <= tsBias**
  - Calendars (Calendars)
  - History (History / Referenced object)

Step 7: Add additional calculated fields in the source file

AssetCenter allows you to add additional fields to your source file. These fields are not saved; just placed in memory.

Use the ⊞, ⊟ and 🔍 icons to add, delete or display these additional fields.

Name

Give a name to this new field.

Formula field type

Enter the way the new field is to be calculated. The aspect of the screen changes according to the type you choose.

Concatenate

This mode lets you combine several fields in the source file. Select these fields one by one. You can separate these fields with the characters of your choice. Simply surround the characters with double quotes ".

Example: Field1" and "Field2.

Fixed extraction

This mode lets you extract part of a text field:

1. Select the source field (called "Main field".)
2. Enter the **Number of characters to ignore**. AssetCenter will skip these characters.
3. Enter the **Number of characters to use**: AssetCenter will keep this number of characters after having skipped the "Characters to ignore".
4. The **Start extraction from end of field** check box tells AssetCenter to skip the **Number of characters to ignore** starting from the end of the field and keeps the **Number of characters to use** starting after the skipped characters and working backwards.

Example:

1. **Number of characters to ignore:** 3
2. **Number of characters to use:** 5
3. Value of field in source file: "REFIMP05A18500"
4. Value imported into the database: "IMP05" if the **Start extraction from end of field** check box was not checked, or "05A18" if it was checked.

Delimited extraction

This mode lets you extract part of a field in the source file:

1. Select the source field (called "Main field".)
2. Indicate the **Separators** used within the values in the main field.

3. Enter the **Number of separators to ignore**. AssetCenter keeps all the data that follows these separators.
4. Enter the **Number of separators to include**. AssetCenter keeps all the information between the start of the text to retain and the separator that follows the last separator to include.
5. Check the **Start extraction from end of field** check box if you want AssetCenter to include the "Number of separators to ignore" and the "Number of separators to include" from the end of the field.

Example:

1. **Separator**: /
2. **Number of separators to ignore**: 2
3. **Number of separators to include**: 3
4. Value of field in source file: "1/2/3/4/5/6/7/8/9"
5. Value imported into the database: "3/4/5/6" if the **Start extraction from end of field** check box was not checked and "4/5/6/7" if it was checked.

Fixed value

This mode lets you include a combination of:

- Character strings enclosed with ".
- Variables. These are certain variables that result from functions used in default field values such as AmLoginName(), AmDate(), AmCounter().

Tree mode

This mode lets you build a tree structure from a single field in the source file.

1. Select the source field (called "Main field".)
2. Indicate the "Separators" used between the sub-values in the field.

AssetCenter divides the source field into sub-values. The number of sub-values equals the number of character strings separated by separation characters. AssetCenter then creates a record for each sub-value and organizes them into a hierarchy.

Example:

1 Create a text file containing a column entitled "Name". The value of the one of the lines in the file is "/France subsidiary/Sales head office/Marketing department".
2 Configure the import module by creating a "Tree mode" type formula field (the separator is "/"). Its name is "FormulaField". Create a "Fixed value" type formula field (value = "1") and map it to the **Department** field (SQL name: bDepartment) field (in order to create departments but not employees).
3 Map "FormulaField" to the **Name** field (SQL name: Name) in the table of employees.
4 Launch the import of the file
5 Result: 3 departments linked hierarchically are created: "France subsidiary", "Sales head office" and "Marketing department".

File

This mode allows you import a file into the database. This useful when importing images or long blocks of text

Files can only be imported into the following types of field:

- Memo,
- Blob.

The following files formats are supported:

- ANSI text,
- Images (all the image formats supported by AssetCenter can be imported).

In the field calculation formula, indicate the source field giving the full pathname of the file to be imported (path, name and extension). By default, the current folder is used as the path.

Script

This mode allows you calculate a value using a Basic script. This script can make reference to imported source fields.

To build the calculation script, either enter the code directly or use click the ▣ button to use the expression builder.

The script cannot reference fields in the database.

Test

Note: This field is only shown when the formula type of the field is set to "Fixed extraction" or "Delimited extraction".

Enter the field value of your choice.

Result

Note: This field is only shown when the formula type of the field is set to "Fixed extraction" or "Delimited extraction".

The simulated import value from the test data appears in this field.

Step 8: Special cases

Importing departments and employees

When importing records from the table departments and employees, it is sometimes necessary to specify if the imported record is a department or an employee.

There is a field that allows you to do this: the **Department** field (SQL name: bDepartment). It is set to "1" for departments and to "0" for employees. By default, the import module considers its value to be "0".

We recommend creating a "Fixed value" type formula field taking the value "1", and attaching this formula to the **Department** field when the imported item is a department.

Note: The import module deduces that a record is a department when this record in turn has a child record in the table of departments and employees; employees cannot have child records.

Importing documents

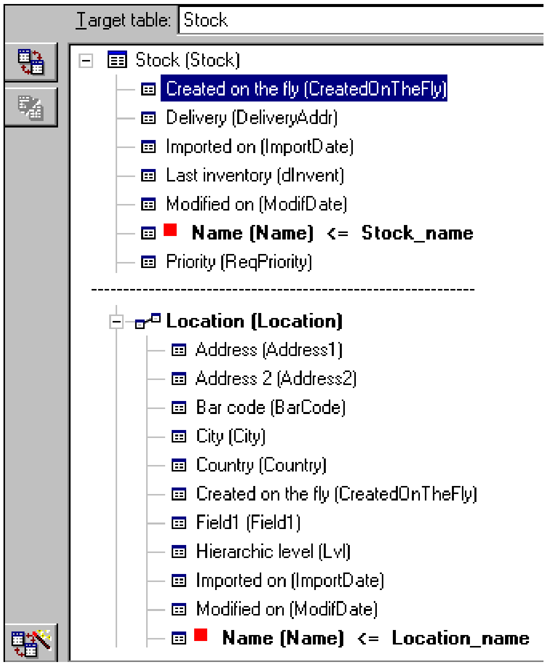
When importing documents you must specify a field to be imported in the **Table** field (SQL name: DocObjTable) in the table of documents. The Table field indicates the SQL name of the table to which the document is linked.

Step 9: Examples of using keys

Here is how AssetCenter interprets your choice of keys:

Example 1: Using a linked field as key of the main table

Figure 15.7. Using a linked field as key of the main table



Target table: Stock

```
Stock (Stock)
    Created on the fly (CreatedOnTheFly)
    Delivery (DeliveryAddr)
    Imported on (ImportDate)
    Last inventory (dInvent)
    Modified on (ModifDate)
    🟥 Name (Name) <= Stock_name
    Priority (ReqPriority)
  ------------------------------------------------------------
  Location (Location)
      Address (Address1)
      Address 2 (Address2)
      Bar code (BarCode)
      City (City)
      Country (Country)
      Created on the fly (CreatedOnTheFly)
      Field1 (Field1)
      Hierarchic level (Lvl)
      Imported on (ImportDate)
      Modified on (ModifDate)
      🟥 Name (Name) <= Location_name
```

In this example, a stock is identified by two main keys:

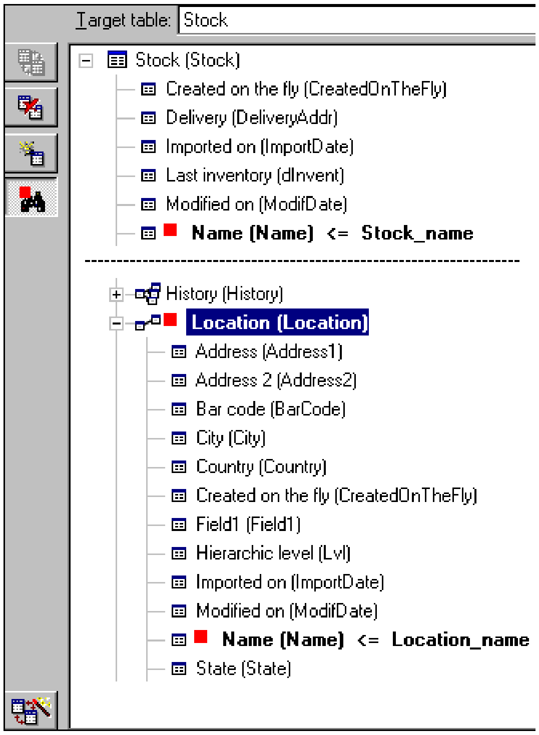
- Stock.Name: Because the 🟥 **Name (Name) <= Stock_name** field is declared as the reconciliation key and it is part of the main table.
- Location.Name: Because the 🟥 **Name (Name) <= Location_name** linked field is declared as the reconciliation key in the table of locations and the 🟥 **Location (Location)** link is declared as an reconciliation key.

In this same example, a location is identified by a main key:

- Location.Name: Because the 🟥 **Name (Name) <= Location_name** field is declared as the reconciliation key in the Locations table.

Example 2: Defining a field as a key in a linked table, even though it is not a key in the main table

Figure 15.8. Defining a field as a key in a linked table



Target table: Stock

```
⊟ Stock (Stock)
    Created on the fly (CreatedOnTheFly)
    Delivery (DeliveryAddr)
    Imported on (ImportDate)
    Last inventory (dInvent)
    Modified on (ModifDate)
    🟥 Name (Name) <= Stock_name
------------------------------------------------------------
  ⊞ History (History)
  ⊟ 🟥 Location (Location)
      Address (Address1)
      Address 2 (Address2)
      Bar code (BarCode)
      City (City)
      Country (Country)
      Created on the fly (CreatedOnTheFly)
      Field1 (Field1)
      Hierarchic level (Lvl)
      Imported on (ImportDate)
      Modified on (ModifDate)
      🟥 Name (Name) <= Location_name
      State (State)
```

In this example, a stock is identified by a single key:

- Stock.Name: Because the 🟥 **Name (Name) <= Stock_name** field is declared as a key and it is part of the main table.
- And the **Location (Location)** link is not a reconciliation key.

In this same example, a location is identified by a key:

- Location.Name: Because the 🟥 **Name (Name) <= Location_name** key is declared as main key.

Conclusion

- You can define keys for the main table and other independent keys for tables linked to the main table. You can thereby import data in several tables from a single text file.
- To declare a field in a linked table as one of the keys in the main table, you must declare the field in the linked table AND the link as identification keys. If you only check the link, the key will only be used as key for the linked table itself.

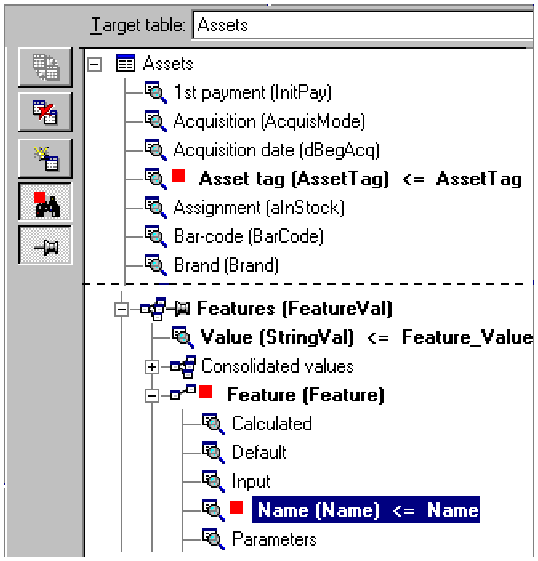
Example 3: Keys that update feature values in database records

To update a record's feature value with a certain value, you must find the (record, feature) pair in the database and assign it the new value. If this pair does not exist, AssetCenter refers to the options defined using the icon for the link, to determine whether or not it should create a linked record.

You can use either of these two methods:

First method (example from the assets table):

Figure 15.9. Key updating values of characteristics



```
Target table: Assets
⊟ Assets
   1st payment (InitPay)
   Acquisition (AcquisMode)
   Acquisition date (dBegAcq)
   ■ Asset tag (AssetTag) <= AssetTag
   Assignment (alnStock)
   Bar-code (BarCode)
   Brand (Brand)
 ⊟ Features (FeatureVal)
    Value (StringVal) <= Feature_Value
  ⊞ Consolidated values
  ⊟ ■ Feature (Feature)
      Calculated
      Default
      Input
      ■ Name (Name) <= Name
      Parameters
```

1 The key on 🟥 Asset tag (AssetTag) identifies the asset to be modified.

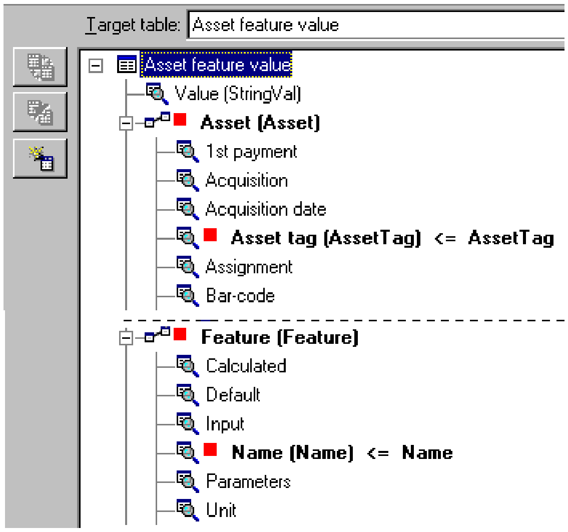
2 The icon on the Features (FeatureVal) * link specifies that you only want to find the features of this asset.

3 The key on 🟥 Feature (Feature) * indicates that the (feature, value) pair is identified by the feature.

4 The key on 🟥 Name (Name) <= Feature.Name * indicates that the feature is identified by its name.

5 The new value appears in the ValString (Value) <= Feature_Value field.

Second method (example from the Asset feature values table):



Target table: Asset feature value

- Asset feature value
  - Value (StringVal)
  - 🟥 **Asset (Asset)**
    - 1st payment
    - Acquisition
    - Acquisition date
    - 🟥 **Asset tag (AssetTag) <= AssetTag**
    - Assignment
    - Bar-code
  - 🟥 **Feature (Feature)**
    - Calculated
    - Default
    - Input
    - 🟥 **Name (Name) <= Name**
    - Parameters
    - Unit

1 The (🟥 Feature (Feature) *, 🟥 Asset (Asset) *) pair identifies the feature associated with the asset.

2 The key on 🟥 Name (Name) indicates which is the key field of the ☒ link.

3 The key on 🟥 Asset tag (AssetTag) indicates which is the key field of the 🟥 Asset (Asset) * link.

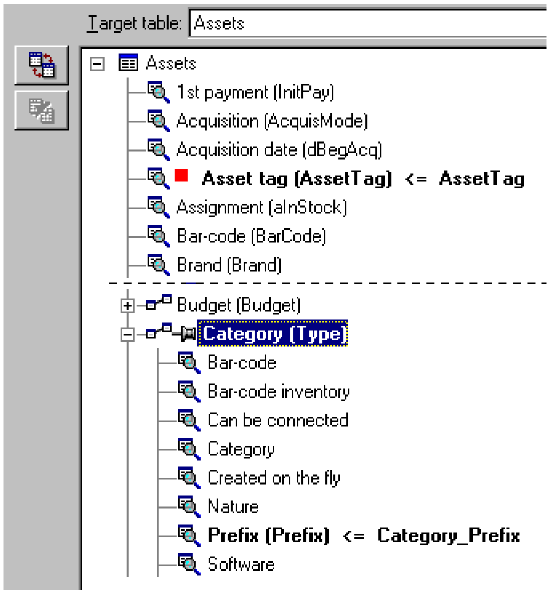
4 The new value appears in the ValString (Value) <= Feature_Value field.

Example 4: Modifying the contents of a linked record for which you have no reconciliation key

Example

Example: You want to modify the prefix of the category that is linked to a given asset. In your import file, you have no key that lets you identify the category. You only know that the category is linked to a given asset.

Figure 15.10. Modifying the content of a linked record for which there is no reconciliation key



Target table: Assets

```
Assets
├─ 1st payment (InitPay)
├─ Acquisition (AcquisMode)
├─ Acquisition date (dBegAcq)
├─ 🟥 Asset tag (AssetTag) <= AssetTag
├─ Assignment (alnStock)
├─ Bar-code (BarCode)
├─ Brand (Brand)
- - - - - - - - - - - - - - - - - - -
├─ Budget (Budget)
├─ Category (Type)
│  ├─ Bar-code
│  ├─ Bar-code inventory
│  ├─ Can be connected
│  ├─ Category
│  ├─ Created on the fly
│  ├─ Nature
│  ├─ Prefix (Prefix) <= Category_Prefix
│  └─ Software
```

1 The key on 🟥 Asset tag (AssetTag) <= Asset.AssetTag * identifies the asset.

2 The symbol on Category (Type) * specifies that you only want to find categories already linked to this asset.

3 The Prefix (Prefix) <= Category_Prefix assigns a new value to the **Prefix** field (SQL name: Prefix) of the category.

Note: To optimize performance, we recommend selecting the keys from among the keys that constitute indexes in the table (Warning: Certain indexes are made up of several fields.)

Step 10: Configuring the transfer

When importing a single text file

Once you have assigned the source fields to fields in the database in step 5, click the button to display the transfer settings window.

When importing a set of text files or an ODBC database

Use the **Edit/ Options** menu item from the import module's main screen.

"Error handling" frame

Select your preferred method for handling errors:

Stop import if error

The import process is halted as soon as an error is encountered.

Commit each imported line

The import module performs a database commit for each imported line. If any errors are detected in the line (either in the main table or in a linked table), the entire line is not imported. The program restores the database to its state before this line was imported.

Commit by group of lines

The import module processes the import by groups of lines; you specify the number of lines per group. If any errors are encountered in the group (either in the main table or in a linked table and for any line of the group), none of the lines in the entire group are imported. The program restores the database to its state before this group was imported. This guarantees database integrity.

"Log file" frame

Select the operations that should appear in the log file:

- Errors
- Appends and updates

Enter the name of the log file and its path. AssetCenter will create the file if it does not exist. Add the extension of your choice. We recommend you use **.log**.

Warning: The import module cannot create directories.

The log file also provides the following information:

- Time the job was dispatched.
- Job description.
- Errors encountered.

The log file is overwritten for each new import procedure.

Step 11: Data transfer

At each of the preceding steps, you can click the Import button to start the data transfer, if enough information is available.

Behavior at the record level

- AssetCenter imports data line by line in the order the data appears in the source file.
- AssetCenter can import data into several different tables from a single line.
- If part of an imported line cannot be imported, AssetCenter creates what it can create.
- AssetCenter searches the AssetCenter database for a record whose identification keys have exactly the same values as the source record. If it such a record is found, AssetCenter modifies it according to the information contained in the text file.

- If no records are found that match the identifier keys, AssetCenter creates a new record in the database.
- If you do not define any reconciliation keys, the import module adds the imported records if the values to be imported respect the unique nature of the basic fields. Without any keys, the import module cannot update records.
- The import module works the same way for main items and linked items.

Behavior at the field level

- The import module does not automatically verify if a field is mandatory. You must check by yourself to make sure that mandatory fields are indeed present in the data to be imported.
- 3If the import module finds an unknown value for a field that is in a "system" itemized list, the line to be imported is rejected.
- If the import module finds a value that is not yet in a standard itemized list, the line to be imported is accepted and the new value is added to the itemized list if the list is "open". If the itemized list is "closed", the line to be imported is rejected.
- If a field value exceeds the maximum length, the end of the value is truncated.
- Fields with an empty value in the text file erase the existing value in the corresponding database field.
- When importing a line of data causes a new record to be created, AssetCenter inserts the default value in fields that do not appear as a column in the text file or the source table. If the column is present but no value is specified, AssetCenter inserts an empty field instead of the default value.

Importing an AssetCenter database

Use the AssetCenter tab in the Import a database option of the File/ Import menu to import an AssetCenter database.

The import procedure is described in detail in the manual entitled "Installation and Upgrade Guide", chapter "Chapter 5 - Upgrading a previous version of AssetCenter", section "Step 1: Converting your previous database" sub-section "Import your previous database into the new AssetCenter 3.0 database".

Saving and executing an import script

A script is a group of import settings saved under a given name. Import scripts let you reproduce similar import conditions without re-defining all the parameters. They help you save time.

Scripts are useful for:

- Executing the same import operation several times, until it works the way you want. (You can modify the source field on each execution, for example.)
- Performing regular database updates (you can update the employees table based on a file from the Human Resources department, for example.)

AssetCenter lets you save scripts and execute them later.

To save a script

When importing a single text file

1. Use the **File/ Import** menu item to select the text file.
2. Define the import settings (name and location of the data file, its structure, etc.)
3. You can click the Save button at any time to save these conditions as a script.

When importing a database or a set of text files

1. Use the **File/ Import** menu item to select the database.

2. Define the import settings (location of the database or text files, mappings between fields, etc.).
3. You can save these settings as a script file at any time using the **File/ Save** or the **File/ Save as** menu item.

To modify a script

When importing a single text file

1. Use the **File/ Import** menu item.
2. Indicate that you are importing a text file.
3. In the "Open data file" dialog box, select "Import scripts *.scr" files in the "list of file types" field.
4. Open the script.
5. Modify the import settings (name and location of the data file, its structure, etc.)
6. You can click the Save button at any time to save these new settings.

When importing a database or a set of text files

1. Use the **File/ Import** menu item to import a database.
2. Populate the **ODBC** tab or the **Text** tab. Click **Open**.
3. Next, open the script using the **File/ Open script** menu item.
4. Modify the import settings (name and location of the database, its structure, etc.)
5. You can save these settings as a script file at any time using the **File/ Save** or the **File/ Save as** menu item.

To create a new script when you are defining import settings

Importing a single text file

Click the Close button. Proceed as you would to create a new script.

Importing a database or a set of text files

Use the **File/ New script** menu item. AssetCenter offers to save the current settings as a script before discarding them.

To execute a script

To execute an existing script:

1. Use the **File/ Import** menu item.
2. Click the icon.
3. Indicate the name of the script file.
4. Start the import procedure.

You may also proceed as if you were going to modify a script and then execute the import procedure from within the import dialog boxes.

Executing an import script from a command prompt

How it works

To execute the DOS program "on line", you need to have created a script beforehand with the Windows import module.

You have the choice between executing it manually or automatically (by using a batch file, for example) an import command using **amimpl32.exe** (located in the **bin32** folder of AssetCenter's program folder).

Syntax

amimpl32 [-verbose] [-?|h|H] -src:<cnx> [-srcpass:<password>] -dst:<cnx> [-dstlogin:<login>] [-dstpass:<password>] [-log:<file>]

-verbose: displays messages during import. Enabled by default.

-?, -h or -H: displays help messages.

-src: according to the case, this parameter shows:

- The path and name of import script to execute.
- AssetCenter database connection name to import completely (as stated in the **File/ Manage connections** menu item, **Name** field).
- The name of an AssetCenter database without connection:

[<EngineName>;<DatabaseLocation>;<User>;<Password>]

In this case, here is how to populate the different fields shown above between <>:

Table 15.4. How to populate the different fields shown above between <>

	Oracle	MS SQL Server	Sybase SQL Anywhere	Sybase SQL Server
EngineName	Oracle	ODBC	ODBC	Sybase
DatabaseLocation	Server name	Data source name	Data source name	ServerName:DatabaseName
User	Account name	MS SQL Server user name	Sybase SQL Anywhere user name	Account name
Password	Account password	MS SQL Server user password	Sybase SQL Anywhere user password	Account password

-srcpass: password associated with the source database to be imported. In the case of an AssetCenter database, this is the password of the "Admin" account.

-dst: AssetCenter database connection name in which the data is to be imported (as stated in the **File/ Manage connections** menu item, **Name** field.)

-dstlogin: login name of AssetCenter account that will receive the imported data ("Admin" account or AssetCenter user with administration rights).

-dstpass: password associated with "dstlogin".

-log: full pathname of the import log file.

Note: If the strings between angle brackets <> contain spaces, you must place them within single quotes (').

Example:

```
amimpl32 -verbose -src:employee.scr
-srcpass:Password -dst:MainDBase -dstlogin:Gerald
 -dstpass:Password -log:'My Log File.txt'
```

16 Exporting data and managing SQL views

CHAPTER

This chapter explains how to export AssetCenter data and manage SQL views of the database.

Definitions

Export scripts

Export scripts allow you to export data, create, recreate or drop SQL view using AssetCenter Export or **amexp32.exe**. Export scripts can be saved in order to be reused later on.

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 AssetCenter Export.

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 start table.
- A list of columns to be extracted (fields, links, features and calculated fields from the start 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 AssetCenter database

You can export data from the AssetCenter database to text files:

- Using an export script.
- Via the **File/ Export list** 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.

Exporting data using an export script

1. Start AssetCenter Export using the **Start** menu or from the AssetCenter program group.
2. Define an export script whose mode is set to "Export data" mode:
   1. In the **Queries** tab, write the queries defining the data to export.
   2. In the **Formatting** tab, specify the format for the text files to which the data will be exported.
   3. Use the **File/ Save script** or **File/ Save script as** menu item to save the script.
3. Execute the export script.
   - Either directly in AssetCenter Export using the <Execute script> ⚙
   - Or by running **amexpl32.exe** from a command prompt.

Note: In order to maintain the integrity of the access restrictions you defined in AssetCenter, only administrators are authorized to start AssetCenter Export or execute amexpl32.exe ("Admin" login or a user with administrative rights).

Exporting data using the File/ Export list menu item

The **File/ Export list** menu item is accessible to all AssetCenter 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 **File/ Export list** menu item.
3. Populate the window that appears, then click the Export button.

Note: For further information on the File/ Export list menu item, please refer to the manual entitled "Reference Guide: "General ergonomics and reference tables", chapter "Using AssetCenter", section"Record lists", sub-section "Exporting a list".

Managing SQL views in the AssetCenter database

AssetCenter Export allows you to create, recreate or delete SQL views in the AssetCenter 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 AssetCenter sense of the term. A SQL view is equivalent to the SQL "CREATE VIEW" statement.

To create, recreate or delete SQL views in the AssetCenter database:

1. Start AssetCenter Export.
2. Define an export script whose mode is set to "Create/Drop SQL views":
    1. In the **Queries** tab, write the queries defining the data to extract.
    2. 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.
    3. Save the export script.
3. Execute the export script:
    - Either directly in AssetCenter Export.
    - Or by running **amexpl32.exe**.

Recommendations

We advise against using the "Id" field in tables as the reconciliation key if you want to subsequently re-import the data you are exporting. In effect, the corresponding ID numbers are not constant, and may be modified. Instead we recommend that you use a key with an unchanging value, e.g. the asset tag.

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 AssetCenter Export.

An export script contains:

- Export queries, which define what should be extracted from the database.
- Formatting options, for exporting data.

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"to (re)create or delete SQL views from the database.

This section explains how to create export scripts:

- Methodology
- Defining export queries
- Output format of an export script
- Actions concerning SQL views

Methodology

To create or modify an export script:

1 Start AssetCenter Export.

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 AssetCenter Export 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 AssetCenter Export.

- 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 **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 AssetCenter 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, 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 column in the list on the right:

- Select the **Visibility** check box in order 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, etc.).
- 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.

  Example:

  ```
  SELECT Brand, Count(lAstId) FROM amAsset GROUP
  BY Brand ORDER BY Brand
  ```

Warning: 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: Except when using SQL Anywhere, you can check the Force indexes option in order to force the use of indexes referenced in the query. For further information, please refer to the manual entitled "Reference Guide: Administration and Advanced Use", chapter "Writing queries in AQL", section "Sorts and indexes".

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 amAsset
```

Example with a check in the **Unique records only** box:

```
SELECT DISTINCT Brand FROM amAsset
```

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 AssetCenter 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** option, 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:
  1. Click the □ button to indicate the name and path of the file.
  2. 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 AssetCenter Export
- Executing an export script from DOS

Executing an export script from AssetCenter Export

To execute an export script from AssetCenter Export:

1. Start AssetCenter Export.
2. Define your export script and save it.
3. Then execute the script:
   - Either by using the **Actions/ Execute the script** menu item.
   - Or by pressing F8.
   - 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:

ℹ General information.

⛔ Error.

❗ Export was successful.

⚠ Warning.

The Cancel button allows you to cancel an export in progress.

Executing an export script from DOS

Principle

In order to execute the DOS software "online", you must first create an export script using AssetCenter Export.

Then you can manually or automatically (e.g. using a batch file) execute an export command using the **amexpl32.exe** program located in the **bin32** subdirectory of the AssetCenter installation directory.

Syntax

```
amexpl32 [-verbose] [-?|h|H] -script:<script>
-cnx:<cnx> [-login:<login>]
[-password:<password>]
```

-verbose: displays messages during the export process.

-?, -h or -H: displays help messages about the software.

-script: path and name of the export file to execute.

-cnx: name of the connection to the AssetCenter 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.

Example:

```
amexpl32 -verbose -script:ibmassets.scx
-cnx:GeneralDatabse -login:Gerald
-password:PAssword
```

17 Using scripts

CHAPTER

This chapter explains how to use scripts.

Definition of a script

Overview

The word "script" generally designates a program written in a high-level language. In AssetCenter, 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 AssetCenter database, etc.

- 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 manuals entitled "Reference Guide: Administration and Advanced Use", chapter "Importing data" and "Reference Guide: Administration and Advanced Use", "Exporting data and managing SQL views"
- "Mixed", both declarative and procedural. This type of script is used in the wizards in AssetCenter.

Information about this version of Basic

The version of Basic used in AssetCenter is a sub-set developed by Cypress compatible with "Visual Basic for ApplicationsTM". Please refer to the Basic documentation for additional information on this language, its structure and syntax.

Only certain Visual Basic for Applications functions are supported, e.g.:

- File access functions are not supported.
- There is limited support for date and time functions, especially under UNIX.
- 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 AssetCenter is similar to standard syntax, except for data access functions from the current record; this uses the following format:

```
 \n \n[Link.Link.Field]
```

Example 17.1. 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 17.2. Example:

```
RetVal=[Contact.Location] or
RetVal=[Contact.lLocaId]
```

Both examples return the same result, the ID of the link.

Applications of scripts

AssetCenter 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 AssetCenter Database Administrator):
  - **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 information on the following:

- Definition of a function

- Built-in functions and programmable functions
- Function and parameter types

Definition of a function

A function is a program that performs operations and returns a value to the user. This value is called 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 AssetCenter.

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 called 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

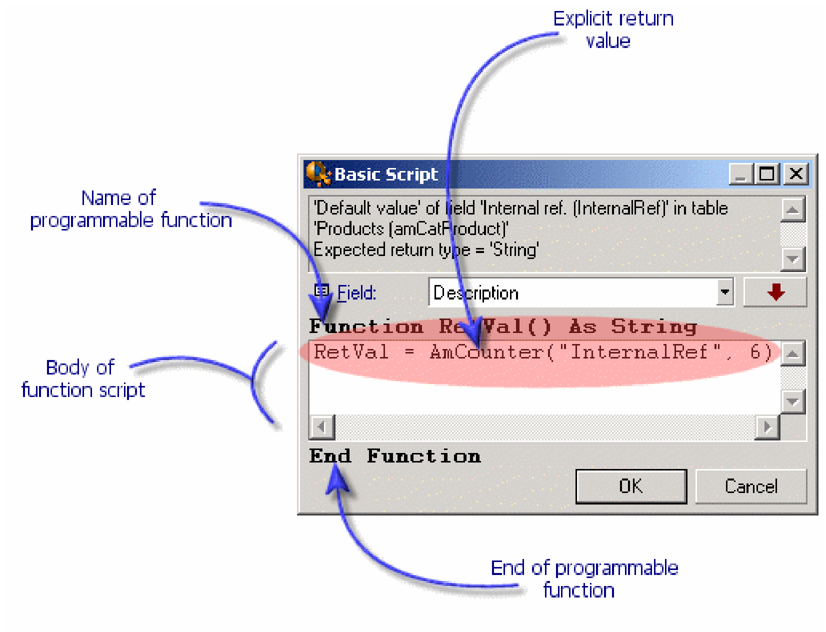
Programmable functions are software items users can write themselves. The user is responsible for explicitly defining the value to be returned in the **RetVal** variable (also called the "return value") by the programmable function, in the following format:

```
RetVal=<Expression>
```

Note: AssetCenter 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:

Figure 17.1. The script builder



'Default value' of field 'Internal ref. (InternalRef)' in table 'Products (amCatProduct)'
Expected return type = 'String'

Field: Description

```
Function RetVal() As String
RetVal = AmCounter("InternalRef", 6)
End Function
```

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. 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+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 AssetCenter 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, AssetCenter 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:

Table 17.1. List of types

Type	Description
Integer	Integer from -32,768 to +32,767.
Long	Integer from -2,147,483,647 to +2,147,483,646.
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 ?" standard
- Generic functions specific to AssetCenter, which may be used in all places where scripts are used.
- Specific functions, which may be used in certain parts of AssetCenter.

First steps in writing scripts

This chapter deals with the functioning of scripts and includes an example scenario:

- Example scenario
- Step 1- Create the Tutorial feature
- Step 2- Open the edit window
- Step 3- Analyze and define the algorithm
- Step 4- Draw up the Basic script

- Step 5- Test the Basic script

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** menu item. Click New to create a new feature. Populate this feature as shown below:

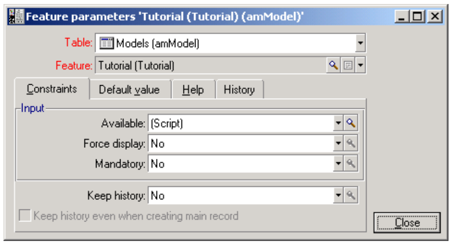
Table 17.2. Fields for populating the feature

Field name	Value
Title (SQL name: TextLabel)	"Tutorial"
SQL name (SQL name: SQLName)	"Tutorial"
Input type (SQL name: seDataType)	**Numerical**

Click Create to create the feature.

Go to the **Parameters** tab and click ⊞ to edit the parameters of the "Tutorial" feature. Populate the **Constraints** tab as shown below.

Figure 17.2. Parameters of the "Tutorial" feature



Feature parameters 'Tutorial (Tutorial) (amModel)'

Table: Models (amModel)

Feature: Tutorial (Tutorial)

Input

Available: (Script)

Force display: No

Mandatory: No

Keep history: No

□ Keep history even when creating main record

Step 2: Open the edit window

In the **Constraints** tab, set the **Available** parameter (SQL name: seAvailable) to "(Script)". Click the magnifier button 🔍. AssetCenter 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/ Models** menu item. Click New to create a new model.

2 Populate the mandatory fields only.

   1 **Name**

   2 **Sub-model of** (SQL name: Parent) with "Computer/Motherboard".

   3 **Nature** (SQL name: Nature)

   4 **Bar code** (SQL name: BarCode).

3 Click **Create** 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.

5 Change the value of the **Sub-model of** field to "/Computer/" and validate this change by clicking **Modify**.

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 library

AssetCenter enables you to save script libraries in order to centralize the access to these scripts.

You can open the script library via the **Tools/ Script library**.

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".

Concepts

In AssetCenter, 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. Populate the **Name** field with the name of your library.
3. Enter your script in the **Script** field.
4. Validate your script by clicking **Create**.

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.

Warning: 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 Departments and employees 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)
```

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.

3 Validate this by clicking **Create**.

Note: The context parameter, normally used with the amEvalSript 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
- Format for Date+Time constants in scripts
- Format for Duration constants in scripts

- Read and write access to a system itemized list
- The CurrentUser virtual link
- Commenting a Basic script
- Triggering an error message

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 AssetCenter. When possible, try to optimize your scripts to maintain overall AssetCenter performance.

Format for "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 and "Unix" date

Dates are expressed differently in Basic and under Unix:

- In Basic, a date can be expressed in international format, or as a floating point number ("Double" type). 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).
- Under Unix, dates are expressed as a long integer ("Long" type) that represents the number of seconds elapsed since January 1, 1870 at midnight, independent of time zones (UTC time).

Format for "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, AssetCenter 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

AssetCenter 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:

Table 17.3. Values used in a system itemized list

Value in itemized list	Associated integer
In use	0
In stock	1
Retired asset	2
Awaiting delivery	3

Thus, in order 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 **Affectation** 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 Infos" sub-folder of the AssetCenter 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.

The "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 user's ID number.
- 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.lPersId]

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. AssetCenter 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; No other precautions are necessary.

```
' 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. In this case, you must use the ":" character to separate the script from the comment. The comment remains prefixed by a single straight quote mark.

```
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>)
```

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.

First example

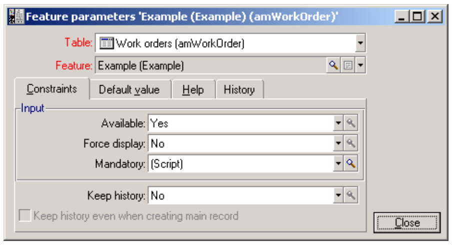
This section deals with a hypothetical problem that can be solved using a Basic script. We recommend trying this problem yourself before consulting the described solution.

- Description of the problem
- Step 1- Analyze and define the algorithm
- Step 2- Draw up the Basic script
- Step 3- Test the Basic script

Description of the problem

A feature called "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:

Figure 17.3. Parameters of the feature "Example1"



Feature parameters 'Example (Example) (amWorkOrder)'

Table: Work orders (amWorkOrder)

Feature: Example (Example)

Input

Available: Yes

Force display: No

Mandatory: (Script)

Keep history: No

□ Keep history even when creating main record

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:

Table 17.4. Analyzing and defining the algorithm - values used in a system itemized list

Value in itemized list	Associated integer
Notified	0
Scheduled	1
In progress	2
Closed	3

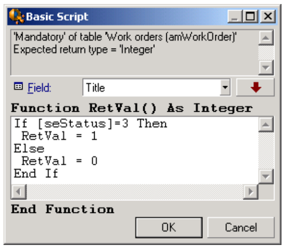
The value in the itemized list that interests us is therefore:

- **Closed** corresponding to the value "3"

Step 2: Draw up the Basic script

You now need to write the algorithm from step 1 in Basic.

Figure 17.4. Editing the program



'Mandatory' of table 'Work orders (amWorkOrder)'
Expected return type = 'Integer'

Field: Title

```
Function RetVal() As Integer
If [seStatus]=3 Then
 RetVal = 1
Else
 RetVal = 0
End If
End Function
```

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 **Maintenance/ Work orders** or **Helpdesk/ 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. AssetCenter 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 displayed but it is an 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
- Step 1- Analyze and define the algorithm
- Step 2- Draw up the Basic script
- 3Step 3- Test the Basic script

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.

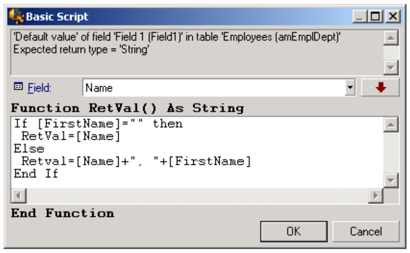
Right-click **Field1** (SQL name: Field1) in the employee detail and select **Configure object**.

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.

Figure 17.5. Editing the program



'Default value' of field 'Field 1 (Field1)' in table 'Employees (amEmplDept)'
Expected return type = 'String'

Field: Name

```
Function RetVal() As String
If [FirstName]="" then
 RetVal=[Name]
Else
 Retval=[Name]+", "+[FirstName]
End If
End Function
```

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 by selecting the **Portfolio/ Departments and employees** menu item and create a new employee.

2 Populate the **Name** (SQL name: Name) and **First** (SQL name: FirstName) fields, then click Create to confirm. AssetCenter displays the name and first name of the employee in **Field1** (SQL name: Field1).

The script fulfills its function correctly.

18 Managing calendars

CHAPTER

Use the Tools/ Calendars menu item to access the list of calendars.

Overview of calendars

The calendar detail

The detail of a calendar shows:

- General information that enables you to uniquely identify the calendar:
  - The **Nom** (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:

- 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.

Note: Warning: 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 AssetCenter.

Impact of calendars on certain areas of functionality

Calendars have an impact on certain areas of functionality of AssetCenter. Modifying a calendar brings about changes both directly and indirectly to certain records in the database. Calendars are involved in:

- 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:

1. Entering general information
2. Populating the Timetable tab
3. Populating the Exceptions tab
4. Checking the calendar

Entering general information

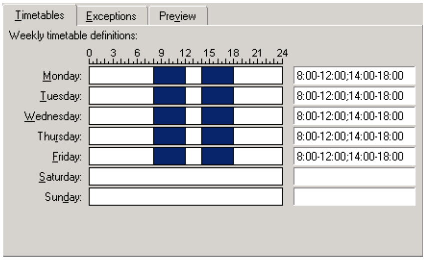
Before entering the actual business hours and exceptions, you must identify the calendar by populating the **Name** field (SQL name: Name) in the detail screen.

You can also associate the calendar with a time zone by populating the **Time zone** field (SQL name: TimeZone).

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.

Figure 18.1. Timetables tab in calendar detail



Weekly timetable definitions:

Monday: 8:00-12:00;14:00-18:00
Tuesday: 8:00-12:00;14:00-18:00
Wednesday: 8:00-12:00;14:00-18:00
Thursday: 8:00-12:00;14:00-18:00
Friday: 8:00-12:00;14:00-18:00
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.
  1. Click the control at the start of the timetable period.
  2. Extend the selection by dragging the mouse to the end of the timetable period. AssetCenter automatically populates the text field to the right of the graduated control.
  3. 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, AssetCenter considers that the 24 hour format is used.

AssetCenter 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.

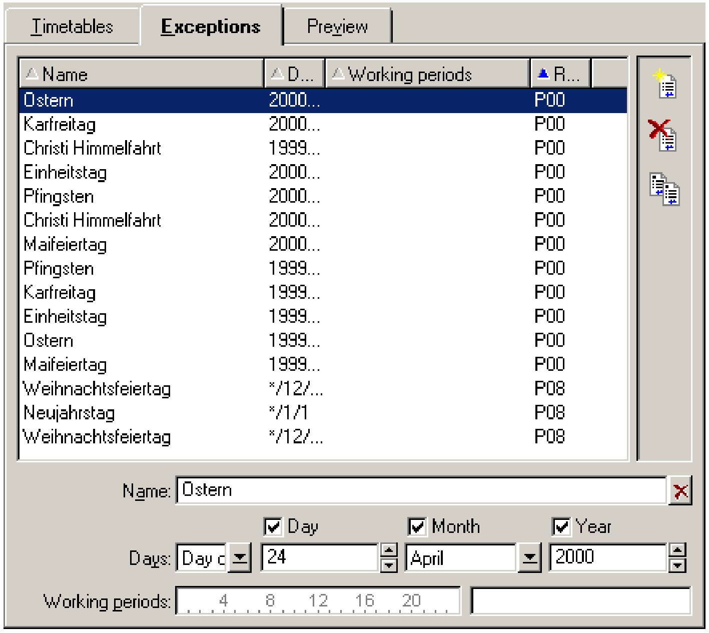
Methodology

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 8:30 A.M. to 10:30 A.M. and from 5:30 P.M. to 6:30 P.M".

The Exceptions tab

Figure 18.2. The Exceptions tab in the detail of a calendar



Name	D...	Working periods	R...
Ostern	2000...		P00
Karfreitag	2000...		P00
Christi Himmelfahrt	1999...		P00
Einheitstag	2000...		P00
Pfingsten	2000...		P00
Christi Himmelfahrt	2000...		P00
Maifeiertag	2000...		P00
Pfingsten	1999...		P00
Karfreitag	1999...		P00
Einheitstag	1999...		P00
Ostern	1999...		P00
Maifeiertag	1999...		P00
Weihnachtsfeiertag	*/12/...		P08
Neujahrstag	*/1/1		P08
Weihnachtsfeiertag	*/12/...		P08

Name: Ostern

☑ Day ☑ Month ☑ Year

Days: Day c 24 April 2000

Working periods: 4 8 12 16 20

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. AssetCenter 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:

Table 18.1. Values of the Days field

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 **Day**, **Month** and **Year** check boxes.
"The first"	Example "The first" Friday of each month. The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **Year** check boxes.
"The second"	Example "The second" Monday of the month of September. The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **year** check boxes.
"The next to last"	Example "The next to last" Wednesday of the month of November. The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **year** check boxes.
"The last"	Example "The last" Tuesday of each month throughout 2000. The rule applies to the weekday defined via the **Day** check box, and for the month(s) and year(s) defined via the **Month** and **Year** check boxes.

Example

Employees at Taltek have the following days off:

- The first Friday of each month is a day off.
- During August, employees at Taltek only work the morning from 8:30 A.M. to 12:30 P.M.

Rule n°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".

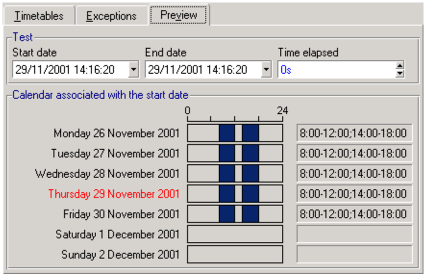
Rule n°2: During August, employees of Taltek only work the morning from 8:30 A.M. to 12:30 P.M.

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 8:30 A.M. to 12:30 P.M. To finish entering the exception you just need to select the period from 8:30 A.M. to 12:30 P.M. in the **Working periods** field.

Checking 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.

Figure 18.3. Preview tab in calendar detail



Start date	End date	Time elapsed
29/11/2001 14:16:20	29/11/2001 14:16:20	0s

Monday 26 November 2001	8:00-12:00;14:00-18:00
Tuesday 27 November 2001	8:00-12:00;14:00-18:00
Wednesday 28 November 2001	8:00-12:00;14:00-18:00
Thursday 29 November 2001	8:00-12:00;14:00-18:00
Friday 30 November 2001	8:00-12:00;14:00-18:00
Saturday 1 December 2001	
Sunday 2 December 2001	

- 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.

19 Managing time zones

CHAPTER

This chapter explains how to use time zones.

Note: This area of functionality is only available for certain AssetCenter license agreements.

Why manage time zones?

Since client machines and the database server can be separated geographically, AssetCenter manages time zones and time differences in relation to Greenwich Mean Time (GMT). AssetCenter respects the following rules:

- All "Date+Time" type fields are displayed on the client machine respecting the time zone of the client machine.
- All "Date+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+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 P.M.
- On the Paris client: May 15, 2000 at 6:30 P.M.
- On the London client; May 15, 2000 at 5:30 P.M.

Implementing time zones

In order for time zones to be handled correctly under AssetCenter you need to follow the following steps:

1. Define the time zones when creating the database under AssetCenter Database Administrator 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/ Options** menu item.

4 Define the calendars in accordance with the time zones.

Creating time zones

Windows 95 and Windows NT can handle time zones and automatically make adjustments for daylight saving changes. AssetCenter manages this information in even greater detail. It takes changes to daylight saving time change rules over the years into account, thus allowing you to display local times in the past with greater accuracy. AssetCenter'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.
- This functionality is not used for locations that do not make changes for daylight saving time.

In order to avoid having to define time zones manually, AssetCenter 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. AssetCenter opens the import selection window.

2 Select "Execute a script" by clicking ⚙. AssetCenter opens the database update screen. Click 🗀 to select the script to execute, in this case **tz.scr** in the **//datakit/standard** sub-folder of the AssetCenter installation folder.

3 Click Import. AssetCenter 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:

- Format of Daylight time field
- Values of the Year argument
- Values of the DaylightInfo argument
- Example

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 called the "parameter"
- <Year> and <DaylightInfo> separately are called "arguments"

The table below gives you an overview of daylight savings changes according to the values of the <Year> and <DaylightInfo> arguments.

Table 19.1. Format of Daylight time field

	The <DaylightInfo> argument is empty	The <DaylightInfo> argument has a value
The <Year> argument is empty. ("<Year>=" does not appear)	There is no change for daylight savings for the entirety of this time zone.	Daylight savings information is valid for all years, excepting those defined by parameters having a <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 (e.g. 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:

Table 19.2. Values of the <DaylightInfo> argument

Subargument	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. 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 describes the time difference between daylight saving time and the time of the time zone concerned.	By default, this sub-argument is set to 60 (which 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" "Tuesday" "Wednesday" "Thursday" "Friday" "Saturday" "Sunday" 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" "February" "March" "April" etc. "November" "December"

Subargument	Description	Possible values
<SDayPos>	Position in the month of the day of change from daylight saving time to standard time.	"First" "Second" "Third" "Fourth" "Last" "Penultimate" (before-last) 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).
<DDay>	Day of change from standard time to daylight saving time.	"Monday" "Tuesday" "Wednesday" "Thursday" "Friday" "Saturday" "Sunday" 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" "February" "March" "April" etc. "November" "December"

Subargument	Description	Possible values
<DDayPos>	Position in the month of the day of change from standard time to daylight saving time.	"First" "Second" "Third" "Fourth" "Last" "Penultimate" (before last) 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, lets 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
```

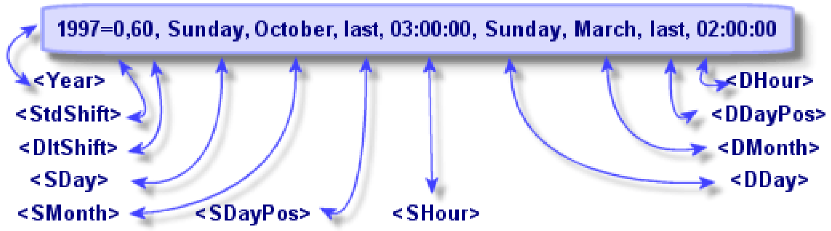
Let's identify the parameters used. The parameters are separated by the pipe character "|". In this case, two parameters are used:

```
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
```

First parameter

Figure 19.1. First parameter



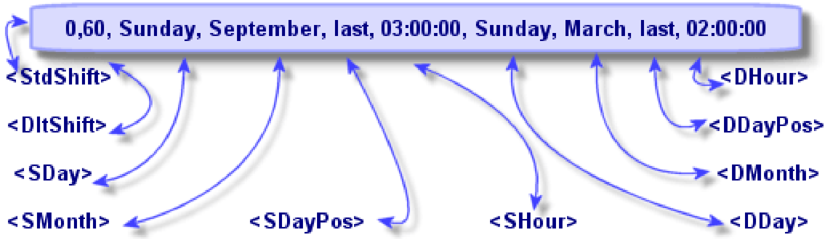
1997=0,60, Sunday, October, last, 03:00:00, Sunday, March, last, 02:00:00

- <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, i.e. 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 3 A.M.
- <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 2 A.M.

Second parameter

Figure 19.2. 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, i.e. 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 3 A.M.

- <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 2 A.M.

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).

Managing time zones in AssetCenter Server

AssetCenter Server allows you to configure tests concerning time zones. Select the **Options/ Configure** menu item.

Tests to perform

In the General tab of the configuration screen, you configure the types of text 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 AssetCenter Server is installed. The time difference is expressed as [(n * 30minutes) + m] where m is between -15 minutes and + 15 minutes.

In both cases

If the time difference exceeds 5 minutes, AssetCenter Server 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 if is the time of the server that requires changing), the connection is refused. You can connect again one 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 AssetCenter Server, is installed).

Specific aspects of the Verify time zone of database server option

Note: To do this, the machine on which AssetCenter Server is running must have the correct time and have the correct information on daylight saving changes.

If necessary, the information on the time zone of the server in the table of options of AssetCenter is updated (if the number (n * 30 minutes) does not correspond to the time zone of the server).

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 AssetCenter, is recovered.

Frequency of the test

The test is performed:

1. First, when AssetCenter Server connects to the database.
2. Then regularly, according to the schedule you define in the **Modules** tab of the configuration screen of AssetCenter Server.

Consequences for various operations

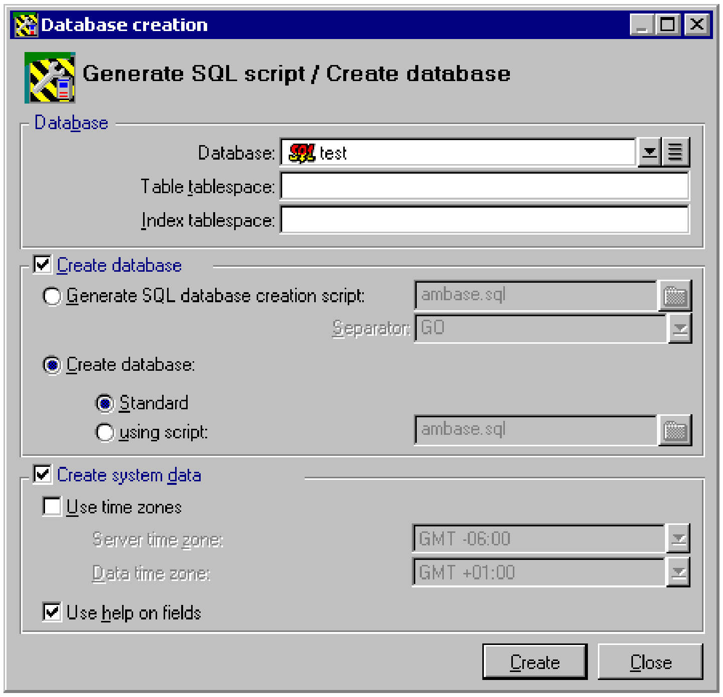
Time zones have an impact on a number of operations:

- Creating the database
- Connecting to a database
- Import and Export

Creating the database

When creating the database, AssetCenter allows you to define time zone options. Select the **Action/ Create database** menu item in AssetCenter Database Administrator. The **Create system data** frame contains the options concerning time zones.

Figure 19.3. Database creation options



Database creation

Generate SQL script / Create database

Database

Database: test

Table tablespace:

Index tablespace:

☑ Create database

○ Generate SQL database creation script: ambase.sql

Separator: GO

● Create database:

● Standard

○ using script: ambase.sql

☑ Create system data

□ Use time zones

Server time zone: GMT -06:00

Data time zone: GMT +01:00

☑ Use help on fields

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 time zone** and **Data time zone** determine the effective time zone of the server at the moment of creation of the database and the time zone according to which the data will be stored.

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, AssetCenter searches in **aam.ini**, (situated in the root folder of windows) for the "LocalTimeZone" entry that defines the time zone of the client machine.

If this information cannot be found, AssetCenter uses the system time zone (defined under Windows NT or Windows 95).

AssetCenter next searches the database for the time zone corresponding to the "LocalTimeZone" entry in **aam.ini**, or the time zone defined in Windows.

The table below summarizes the different possible situations:

Table 19.3. Different possible situations for the LocalTimeZone when connecting to a database

"LocalTimeZone" exists in aam.ini?	Corresponding time zone found in table of time zones?	Information saved in aam.ini ("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, AssetCenter 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. AssetCenter 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 (i.e. 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), AssetCenter proposes adjusting clock time at the client level.

Connection fails if the user refuses.

AssetCenter 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_lTimeZoneCheckInMns** option in **aam.ini** (situated in the root folder of Windows), in the [option] section.

```
[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 AssetCenter.

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+Time" fields correspond to the time zone of the machine carrying out the import or export.

20 Using AssetCenter as a DDE server

CHAPTER

This section aims to describe in detail the DDE calls recognized by AssetCenter when used as a DDE server.

In this section, each theoretical description is followed by a practical example.

Definition of a DDE server

DDE means Dynamic Data Exchange and designates dynamic data exchange mechanisms between Windows applications. In the case being described, DDE is used to execute AssetCenter commands from another application

DDE call mechanisms

The DDE mechanisms are based on "services". In order to execute a DDE mechanism, you need to define a "topic" to give the necessary context in which the "commands" are to be executed. For reasons of integrity, each time you change context, you must terminate the previous context.

This chapter contains information on the following topics:

- DDE service
- DDE topic
- DDE command

DDE service

In most cases, the "service" is the name of the executable loaded in memory. In this case, i.e. when using AssetCenter as a DDE server, the service is "aam".

DDE topic

The topic allows you to define the context in which the action is to be executed. For AssetCenter, this topic is "AssetCenter".

DDE command

These are commands that will be sent to AssetCenter to be executed. These can be divided into several groups:

- Global commands, which do not require a table name or field name to be executed.
- Commands associated with a table, requiring the SQL name of a table as a parameter in order to be executed.
- Commands associated with a table and field or link, requiring the SQL name of a table and field or link as parameters in order to be executed.

Commands belonging to these groups can be of two types:

- "Execute", which enables you to execute a task in AssetCenter.
- "Request", which enables you to ask AssetCenter for information.

How to find the SQL name of a table, field or link

When you right-click any field of a table, AssetCenter displays a shortcut menu. If you select the **Configure object** menu item, the window displayed by AssetCenter gives the SQL name the table and SQL name of the object (link or field) on which you have clicked.

Introduction to DDE commands

Steps to follow

Three steps are necessary for the proper execution of a DDE command:

1 Clearly define the context of execution of the command by specifying the "Service" and the "Topic" to be used. When using AssetCenter as a DDE server, the "Topic" is always "AssetCenter".

> Note: Once the context is defined, it is used by default in all following DDE commands until a new context is defined.

2 Launch the command itself. Two types of command are available.

- **Execute: <command>(<parameters>)**
- **Request: <command>(<parameters>)**

3 Close the previously defined context.

Particularities

Each application under Windows has its own way of sending and receiving DDE commands. The following sections of this document include:

- An exhaustive list of DDE commands that AssetCenter can receive. The syntax of each of these commands is described in detail.

- Examples of how AssetCenter can be driven by DDE mechanisms, which illustrate the use of these commands. Each of these examples, uses a different programming language.

Global commands

"Global" commands do not depend on a given table or field. In particular, they do not require the SQL name of a table or a field as an argument.

This chapter contains information on the following commands:

- Connect(Cnx, User, Password)
- Disconnect()
- ExecuteAction(ActionName)
- ListAllTables([Mask])
- ListOpenTables([Mask])
- OpenView(ViewName)

Connect(<Cnx>, <User>, <Password>)

Type of action

Execute

Description

Connects to a database using the following parameters:

<Cnx>

This argument can either contain:

- A connection name defined under AssetCenter (which can be found in the **amdb.ini** file).
- The full definition of a connection to a database according to the syntax described below:

```
[<Database engine>;<Database localization>;<Login
 for the database engine>;<Password for the
database engine>]
```

<User>

This argument contains the name of the AssetCenter user used to connect to the database.

<Password>

This argument contains the password associated with the login (value of "<User>" argument).

Examples

The following command allows you to connect to an Oracle database for which a connection is already defined under AssetCenter. The name of this connection is "TDemo". The "Admin" login is used for the connection. The password is "password".

```
Execute:Connect(TDemo, Admin, password)
```

The following command performs the same connection without the connection being defined under AssetCenter. The "TDemo" database is located on the server "Joshua". The password for the Oracle connection is "Root".

```
Execute:Connect([Oracle;Joshua;TDemo;Root], Admin,
 password)
```

Disconnect()

Type of action

Execute

Description

Disconnects AssetCenter from the current database.

Example

The following command terminates the connection to the AssetCenter database:

```
Execute:Disconnect()
```

ExecuteAction(<ActionName>)

Type of action

Execute

Description

Triggers the action called "<ActionName>".

<ActionName>

This argument contains the name of the action as defined under AssetCenter in the **Name** field (SQL name: Name) of the action detail.

Example

The following command triggers the action "Reminder: work order not completed":

```
Execute:ExecuteAction(Reminder: Work order not
completed)
```

ListAllTables([Mask])

Type of action

Request

Description

Lists all tables in the database. This list, made up of the SQL names of tables, can be filtered by using the "<Mask>" argument.

<Mask>

This argument allows you to filter data using the following:

- A question mark ("?") can be used as a wildcard for any single character.
- An asterisk ("*") can be used to represent any character or group of characters.

Examples

The following command gives the list of SQL names of all the tables in the current database:

```
Request:ListAllTables()
```

The following command gives the list of SQL names of all the tables in the current database whose SQL names start with "amA":

```
Request:ListAllTables(amA*)
```

The following command gives the list of SQL names of all the tables in the current database containing the letter "v":

```
Request:ListAllTables(*v*)
```

The following command gives the list of SQL names of all the tables in the current database starting with "am" and whose fourth character is "t":

```
Request:ListAllTables(am?t*)
```

ListOpenTables([Mask])

Type of action

Request

Description

Lists the SQL names of all open tables of the database. This list can by filtered using the "<Mask>" argument.

<Mask>

This argument allows you to filter data using the following:

- A question mark ("?") can be used as a wildcard for any single character.
- An asterisk ("*") can be used to represent any character or group of characters.

Examples

The following command lists the SQL names of all open tables in the current database:

```
Request:ListOpenTables()
```

The following command lists the SQL names of all open tables in the current database whose SQL name starts with "amA":

```
Request:ListOpenTables(amA*)
```

Let's suppose that the three following tables are open under AssetCenter: "amAsset", "amAction", "amModel". The previous command returns the SQL name of the two tables beginning with "amA", i.e.: "amAsset" and "amAction".

OpenView(<ViewName>)

Type of action

Execute

Description

Opens a view defined under AssetCenter.

<ViewName>

This argument contains the SQL name of the view as it is defined under AssetCenter.

Example

The following command opens the view called "Leased assets":

```
Execute:OpenView(Leased assets)
```

Commands associated with a table

These commands depend on a table. They require the SQL name of a table as an argument in order to be executed.

This chapter includes information on the following commands:

- OpenTable(Table)
- CloseTable(Table)
- Table.GetRecordCount()
- Table.SetViewMode(Mode)
- Table.SetRecordMode(Mode)
- Table.ListAllFields([Mask])
- Table.ListAllLinks([Mask])
- Table.SetFilter(Condition)
- Table.SetSelection(Condition)
- Table.GetSelectionId()

OpenTable(<Table>)

Type of action

Execute

Description

Opens the table with SQL name "<Table>".

<Table>

This argument contains the SQL name of the table to be opened.

Example

The following command opens the table of assets (SQL name: amAsset):

```
Execute:OpenTable(amAsset)
```

CloseTable(<Table>)

Type of action

Execute

Description

Closes the table previously opened under AssetCenter.

<Table>

This argument contains the SQL name of the table to be closed.

Example

The following command closes the table of assets (SQL name: amAsset):

```
Execute:CloseTable(amAsset)
```

<Table>.GetRecordCount()

Type of action

Request

Description

Returns the number of records in the table with SQL name "<Table>". The table concerned by this command must first be opened in order for this command to work.

<Table>

This argument contains the SQL name of the table for which you want to know the number of records.

Example

The following command returns the number of records in the table of asset (SQL name: amAsset):

```
Request:amAsset.GetRecordCount()
```

<Table>.SetViewMode(<Mode>)

Type of action

Execute

Description

Defines the view mode of a table that has been opened.

<Table>

This argument contains the SQL name of the table concerned.

<Mode>

This argument can be set to one of the following values:

- "Arbo": Records in the table "<Table>" are displayed in tree view.
- "List": Records in the table "<Table>" are displayed in list view.
- "ListOnly": Records from the table "<Table>" are displayed only.
- "DetailOnly": Only the detail of the selected record in the table "<Table>" is displayed.
- "ListDetail": Displays both the list of records in the table "<Table>" and the detail of the selected record in this list.

Example

This following command puts the table of assets (SQL name: amAsset) in tree-view:

```
Execute:amAsset.SetViewMode(Arbo)
```

<Table>.SetRecordMode(<Mode>)

Type of action

Execute

Description

Defines the mode of interaction with records from an open table.

<Table>

This argument contains the SQL name of the table concerned.

<Mode>

This argument can be set to one of the following values:

- "New": Starts the creation of a new record in the table "<Table>". The command corresponds to clicking the New button.
- "Duplicate": Duplicates the selected record in the table "<Table>". The command corresponds to clicking the Duplicate button.
- "Delete": Destroys the record selected in the table "<Table>". The command corresponds to clicking the Delete button.
- "Modify": Confirms the modifications made to the selected record in the table "<Table>". The command corresponds to clicking the Modify button.
- "Create": Confirms the creation of a new record in the table "<Table>". The command corresponds to clicking the Create button.
- "CreateContinue": Combines a creation and a duplication. The command corresponds to clicking the Create button.
- "Cancel": Cancels the creation of a new record or the modifications made to the selected record. The command corresponds to clicking the Cancel button.
- "Close": Closes the previously opened table "<Table>". The command corresponds to clicking the Close button.

Example

The following example opens the table of assets (SQL name: amAsset), starts the creation of a new record, then cancels this creation:

```
Execute:OpenTable(amAsset)
Execute:amAsset.SetRecordMode(New)
Execute:amAsset.SetRecordMode(Cancel)
```

<Table>.ListAllFields([Mask])

Type of action

Request

Description

Returns the SQL names of all the fields in the previously opened table "<Table>".

<Table>

This argument contains the SQL name of the table concerned.

<Mask>

This argument allows you to filter data using the following:

- A question mark ("?") can be used as a wildcard for any single character.
- An asterisk ("*") can be used to represent any character or group of characters.

Example

The following command returns the SQL names of all the fields in the table of assets:

```
Request:amAsset.ListAllFields
```

The following command returns the SQL names of all the fields in the table of assets (SQL name: amAsset) whose SQL name starts with "se":

```
Request:amAsset.ListAllFields(se*)
```

<Table>.ListAllLinks([Mask])

Type of action

Request

Description

Returns the SQL names of all the links in the previously opened table "<Table>".

<Table>

This argument contains the SQL of the table concerned.

<Mask>

This argument allows you to filter data using the following:

- A question mark ("?") can be used as a wildcard for any single character.
- An asterisk ("*") can be used to represent any character or group of characters.

Example

The following command returns the list of SQL names of all the links of the table of assets (SQL name: amAsset):

```
Request:amAsset.ListAllLinks
```

The following command returns the list of SQL names of all the links of the table of assets (SQL name: amAsset) whose SQL name starts with "se":

```
Request:amAsset.ListAllLinks(se*)
```

<Table>.SetFilter(<Condition>)

Type of action

Execute

Description

Filters the table "<Table>" according to the "<Condition>" argument.

<Table>

This argument contains the SQL name of the table concerned.

<Condition>

This argument contains the condition applied to the command. It is an AQL clause.

Example

The following command filters the table of assets (SQL name: amAsset). This filter displays only those records modified before 8/28/98 at 15:00:00:

```
Execute:amAsset.SetFilter(dtLastModif<[98/08/28
15:00:00])
```

<Table>.SetSelection(<Condition>)

Type of action

Execute

Description

Selects one or more records from the previously opened table "<Table>", according to the argument" <Condition>".

<Table>

This argument contains the SQL name of the table concerned.

<Condition>

This argument contains the condition applied to the command. It is an AQL clause.

Example

The following command selects assets whose asset tag is greater than or equal to "7":

```
Execute:amAsset.SetSelection(AssetTag>='7')
```

<Table>.GetSelectionId()

Type of action

Request

Description

Returns the list of identifiers of selected records in the table "<Table>".

<Table>

This argument contains the SQL name of the table concerned.

Example

The following example selects those records in the table of assets (SQL name: amAsset) whose asset tag is greater than or equal to "7", then returns the list of identifiers of the selected records:

```
Execute:amAsset.SetSelection(AssetTag>='7')
Request:amAsset.GetSelectionId()
```

Commands associated with a table and a field or a link

These commands depend on a field in a table. They require the SQL name of a table and the SQL name of a field or a link in this table as argument in order to be executed.

This chapter contains information on the following commands:

- Table-Objet.AddLink()
- Table-Object.GetValue()
- Table-Object.Highlight()
- Table-Object.RemoveLink()
- Table-Object.SetFocus()
- Table-Object.SetValue(Value)
- Table-Link.SetValueWhere(Condition)

- Table-Object.Show()

<Table>:<Object>.AddLink()

Type of action

Execute

Description

Simulates clicking the ⊞ button of a list zone. This command allows you to add a link to a record in a table.

<Table>

This parameter contains the SQL name of the table concerned by the operation.

<Object>

This parameter contains the SQL name of the object concerned.

Example

The following command adds a value in an itemized list.

```
Execute:amItemizedList:ItemListVals.AddLink()
```

<Table>:<Object>.GetValue()

Type of action

Request

Description

Returns the value of an "<Object>" (field or link) of the table "<Table>" for the selected record.

<Table>

This argument contains the SQL name of the table concerned.

<Object>

This argument contains the SQL name of the field or link of the table "<Table>" for which you want to recover the value.

Example

The following command returns the values of the **Field1** field (SQL name: Field1) in the table of assets (SQL name: amAsset):

```
Request:amAsset:Field1.GetValue()
```

The following command returns the value of the **Category** link (SQL name: Category) of the table of assets (SQL name: amAsset):

```
Request:amAsset:Category.GetValue()
```

The following command returns the value of the **Remarks** link (SQL name: Comment) of the table of assets (SQL name: amAsset):

```
Request:amAsset:Comment.GetValue()
```

<Table>:<Object>.Hilight()

Type of action

Execute

Description

Sets the focus on a field and highlights it.

<Table>

This parameter contains the SQL name of the table concerned by the operation.

<Object>

This parameter contains the SQL name of the field concerned by the operation. This command does not work for links.

Example

The following command highlights the **Bar code** field (SQL name: Barcode) in the table of assets (SQL name: amAsset):

```
Execute:amAsset:Barcode.Hilight()
```

<Table>:<Object>.RemoveLink()

Type of action

Execute

Description

Simulates clicking the ⊟ button of a list zone. This command allows you to delete a link to a record in a table.

<Table>

This parameter contains the SQL name of the table concerned by the operation.

<Object>

This parameter contains the SQL name of the object concerned.

Example

The following command deletes the selected value in an itemized list.

```
Execute:amItemizedList:ItemListVals.RemoveLink()
```

<Table>:<Object>.SetFocus()

Type of action

Execute

Description

Sets the focus on the field or link "<Object>" of the table "<Table>" for the selected record.

<Table>

This argument contains the SQL name of the table containing the field or link to which you want to set the focus.

<Object>

This argument contains the SQL name of the field or link or the table "<Table>" to which you want to set the focus.

Example

The following command sets the focus on the **Category** link (SQL name: Category) of the table of assets (SQL name: amAsset):

```
Request:amAsset:Category.SetFocus()
```

<Table>:<Object>.SetValue(<Value>)

Type of action

Execute

Description

Populates the field "<Field>" in the table "<Table>", for the selected record, with the value "<Value>".

<Table>

This argument contains the SQL name of the table containing the field that you want to populate.

<Field>

This argument contains the SQL name of the field in the table "<Table>" that you want to populate.

<Value>

This argument contains the value you want to assign to the "<Field>" field in the table "<Table>" for the selected record. In the case of a "Date" or "Date+Time" type field, this argument must be expressed using the international date format (yy/mm/dd hh:mm:ss).

Example

The following command attributes the value "Test" to the **Field1** field (SQL name: Field1) in the table of assets (SQL name: amAsset), for the selected record:

```
Execute:amAsset:Field1.SetValue(Test)
```

The following command attributes the value "08/28/00" to the **Install. date** field (SQL name: dInstall) in the table of assets (SQL name: amAsset), for the selected record:

```
 Execute:amAsset:dInstall.SetValue(00/08/28)
```

<Table>:<Link>.SetValueWhere(<Condition>)

Type of action

Execute

Description

Populates the link "<Link>" in the table "<Table>", for the selected record, according to the condition "<Condition>".

<Table>

This argument contains the SQL name of the table containing the table you want to populate.

<Link>

This argument contains the SQL name of the link of the table "<Table>" that you want to populate.

<Condition>

This argument allows you to identify the target record of the link. It is an AQL clause.

Example

The following example assigns the value "Test" to the **Category** link (SQL name: Category) from the table of the assets (SQL name: amAsset) for the selected record. The category "Test" must exist in order for the DDE command to execute correctly.

```
Execute:amAsset:amCategory.SetValueWhere(Name='Test')
```

<Table>:<Object>.Show()

Type of action

Execute

Description

Moves the focus to a field or a link that is not visible on the screen. The table containing the field or link must be open.

<Table>

This parameter contains the SQL name of the table concerned by the operation.

<Object>

This parameter contains the SQL name of the object concerned.

Example

The following command moves the focus to the **A/C Code** field (SQL name: AcctCode) in the table of assets (SQL name: amAsset):

```
Execute:amAsset:AcctCode.Show()
```

Introduction to Examples of DDE calls

In order to illustrate this area of functionality, we have described several scenarios.

- The first scenario gives a theoretical description of DDE calls.
- The second is a practical example of using DDE calls under Excel. The associated example is written in Visual Basic for Applications.

- The third scenario presents an application written in Visual Basic 5.0 that will enable you to gain some hands-on experience with DDE calls.

First scenario: AssetCenter internal DDE calls

The objective of this scenario is to identify, for each one of the actions to be carried out, the appropriate DDE command and arguments. This scenario deals with the theoretical aspects of using DDE mechanisms; the practical aspects are dealt with in the following examples.

In this example, we are going to create a "numerical" type feature, called "RAM". Here are the actions carried out:

1. Opening the table of features
2. Entering the title of the feature
3. Entering the data entry type of the feature
4. Entering the unit
5. Creating the feature

Answering the call

We must first identify the service and topic needed to execute the DDE commands.

We are in the general context of the AssetCenter application, therefore:

- Service = "aam"
- Topic: "AssetCenter"

We now just have to execute the command that opens the table of features:

- Command: OpenTable()
- Parameter: The SQL name of the table; in this case "amFeature"

The command is written as follows:

```
OpenTable(amFeature)
```

AssetCenter opens the table of features. We now need to initiate the creation of a new record for this table:

- Command : SetRecordMode()
- Prefix of the command: The SQL name of the table, i.e. "amFeature"
- Parameter: The date entry mode; in this case "New"

The command is written as follows:

```
amFeature.SetRecordMode(New)
```

Entering data

Next, we need to give AssetCenter the appropriate commands for the fields that interest us:

- **Title** field (SQL name: TextLabel).The command to use and its arguments are as follows:
    - Command : **<Table>:<Object>.SetValue(<Value>)**

      * Argument "<Table>: The SQL name of the table, i.e. "amFeature"

      * Argument "<Object>": The SQL name of the field, i.e. "TextLabel"

      * Argument "<Value>": The value of the field, i.e. "RAM"

      ```
      amFeature:TextLabel.SetValue(RAM)
      ```
- **Input type** field (SQL name: seDataType). The command to use and its arguments are as follows:
    - Command: **<Table>:<Object>.SetValue(<Value>)**

      * Argument "<Table>: The SQL name of the table, i.e. "amFeature"

      * Argument "<Object>": The SQL name of the field, i.e. "seDataType"

      * Argument "<Value>": The value of the field, i.e. "Numerical"

      ```
      amFeature:seDataType.SetValue(Numeric)
      ```
- **Unit** field (SQL name: Unit). The command to use and its arguments are as follows:
    - Command: **<Table>:<Object>.SetValue(<Value>)**

* Argument "<Table>: The SQL name of the table, i.e. "amFeature"

* Argument "<Object>": The SQL name of the field, i.e. "Unit"

* Argument "<Value>": The value of the field, i.e. "MB"

```
amFeature:Unit.SetValue(MB)
```

Creating the feature

In order to create the feature, we just need to create the record in the table of features:

- Command : <Table>.SetRecordMode(<Mode>)
  - Argument "<Table>": The SQL name of the table, i.e. "amFeature"
  - Argument "<Mode>": The mode of creation of a record, i.e. "Create"

```
amFeature.SetRecordMode(Create)
```

Second scenario: DDE calls from Excel

This example is based on an Excel worksheet "TestDDE.xls", which can be found in the **amdemo** folder of AssetCenter. This file contains the necessary macros for this example.

This example is identical to the previous one, only the DDE calls are different and are in conformity with VBA (Visual Basic for Applications).

This section contains the following information:

- Description of the macro
- Source code of macro

Description of the macro

Go to the "Data_entry" worksheet. It contains a table with three columns named **Title**, **Input type**, and **Unit** respectively, and a button (labeled "Create the feature") to which a macro called "Create" has been assigned:

You just need to enter the appropriate values in the **Title** (cell B6 in this example), **Input type** (cell C6), **Unit** (cell D6) fields, and click "Create the feature" in order for Excel to give issue the orders to AssetCenter to perform the following actions:

1. Opening the table of features
2. Entering the title of the feature
3. Entering the data entry type of the feature
4. Entering the unit
5. Creating the feature

Source code of macro

For reference, here is the source code of the macro that executes the DDE calls. For further information on DDE calls under Excel or Word and the syntax of VBA language, please refer to the respective online documentation.

```
Sub CreateFeature()
Set Label = Worksheets("Data_entry").Range("B6")
Set Type = Worksheets("Data_entry").Range("C6")
Set Unit = Worksheets("Data_entry").Range("D6")
Context = Application.DDEInitiate(app:="aam",
topic:="AssetCenter")
Application.DDEExecute Context,
"OpenTable(amFeature)"
Application.DDEExecute Context,
"amFeature.SetRecordMode(New)
Application.DDEExecute Context,
"amFeature:TextLabel.SetValue("+Label+")"
Application.DDEExecute Context,
"amFeature:seDataType.SetValue("+Type+")"
Application.DDEExecute Context,
"amFeature:Unit.SetValue("+Unit+")"
Application.DDEExecute Context,
"amFeature.SetRecordMode(Create)"
```

```
Application.DDETerminate Context
End Sub
```

Third scenario: DDE calls from Visual Basic

This scenario presents a utility that enables you experiment with DDE mechanisms via a simple graphical interface. It is also a good illustration of how to program DDE mechanisms under Visual Basic.

For this example, execute **DDE TestCenter.exe**, which can be found by following this path: **samples\DDE\Program**. This program allows you to execute "Execute" and "Request" type DDE commands.

Note: AssetCenter must be launched in order to be able to receive DDE commands issued by the Basic program.

This chapter contains information on the following topics:

- Before starting
- Executing the program
- Source code of the program

Source code of the program

The commented source code to this program, in the form of a Visual Basic project, can be found by following this path: **sample\DDE\VisualBasic**.

Before starting

Installation

In order to use this program, Visual Basic must be correctly installed on your machine. In particular, certain ActiveX controls must be correctly registered. If DDE TestCenter returns an error such as "Control XXXX is not registered", try the following:

1. Search for the control on your computer and go to its folder.
2. Execute the following command

   ```
   regsvr32 XXXX
   ```

3. Relaunch DDE TestCenter. If this fails, please refer to the documentation of Visual Basic.

Recommendations

In order to make the most of this example, we recommend that you:

1. Launch AssetCenter, then reduce the size of the application window to about half the size of your screen.
2. Launch "DDE TestCenter.exe" and move the application window so that is next to that of AssetCenter.

Note: In this way, you will be able to see the results of the orders issued by DDE TestCenter.exe in AssetCenter directly.

Syntax

This example is similar to the previous one, only the DDE calls are different and correspond to the Visual Basic standard.

Executing the program

"Execute" type DDE command

Go to the "Execute" tab:

Enter the command to be executed in the "Command" field. Use the following syntax:

```
Command=<command>(<arguments>)
```

Click the button to execute the command. Any errors are displayed in the "Last DDE Error" field.

Example n°1:

The following "Execute" command opens the table of features:

```
OpenTable(amFeature)
```

Example n°2:

The following "Execute" commands open the table of budgets, create a new record and populate the "Name" field (SQL name: Name) in the detail screen. Execute this commands sequentially:

```
OpenTable(amBudget)
amBudget.SetRecordMode(New)
amBudget:Name.SetValue("Test")
```

"Request" type DDE command

Go to the "Request" tab:

Enter the command to be executed in the "Command" field. Use the following syntax:

```
Command=<command>(<arguments>)
```

Click the button to execute the command. The results of the request are displayed in the "Request Result" field. Any errors are displayed in the "Last DDE Error" field.

Example n°1:

The following "Request" command displays the list of SQL names of all the tables in the current connection:

```
ListAllTables()
```

Example n°2:

The following "Request" command displays the list of SQL names of all the fields of the table of features (SQL name: amFeature) opened beforehand:

```
amFeature.ListAllFields()
```

21 Calculated fields

CHAPTER

Use the Administration/ Calculated fields menu item to access the calculated field creation screen.

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 AssetCenter 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 AssetCenter 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
- Methodology used to create calculated fields

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:

Table 21.1. 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 Integrated editor	Limited language. Fields of this type cannot be 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 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 field.

An AQL query can make use of all three properties (can be displayed, sorted, used in filters):

Table 21.2. Correspondences

Property	Corresponding AQL arguments
Can be displayed	`SELECT` clause

Can be sorted	`SELECT`
	`ORDER BY`
	`GROUP BY`
	clauses

Can be used in filters	`SELECT`
	`ORDER BY`
	`GROUP BY`
	`WHERE`
	`HAVING`
	clauses

For further information on AQL queries, refer to the manual entitled "Reference Guide: Administration and Advanced Use" chapter "Writing queries in AQL".

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.

Methodology used to create calculated fields

This chapter 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:

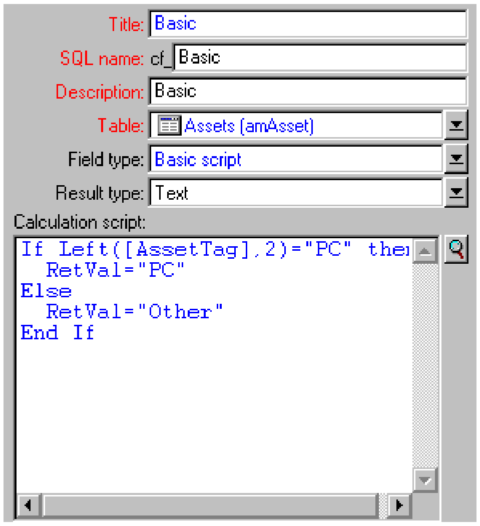
- If the field is for informational purposes only, a Basic type calculated field will suffice.
- If you want to be able to sort records according to the value of the field, "AQL" or "Calculated string" must be used.
- If you want to be able to filter records according to the value of the field, "AQL" must be used.

Once you have defined your needs, you can move on to the next phase.

Open the creation screen

Select the **Administration/ Calculated fields** menu item. AssetCenter displays the calculated field creation screen:

Figure 21.1. Detail of a calculated field



Title: Basic

SQL name: cf_ Basic

Description: Basic

Table: Assets (amAsset)

Field type: Basic script

Result type: Text

Calculation script:

```
If Left([AssetTag],2)="PC" the
  RetVal="PC"
Else
  RetVal="Other"
End If
```

Identify the calculated field

First, populate the upper part of this screen in order 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, please refer to the following documentation:

- Manual entitled "Reference manual: Administration and advanced use", chapter "Using scripts" for the Basic language. The function used is **RetVal()**.
- Manual entitled "Reference manual: Administration and advanced use", chapter "Writing queries in AQL" for the AQL language.
- Manual entitled "Reference manual: Administration and advanced use", chapter "Structure of the AssetCenter database", paragraph

"Description of the tables", sub-paragraph "AssetCenter table description strings" for calculated strings.

Define the user rights for the calculated field

Select the **Administration/ Rights management/ User rights** menu item. AssetCenter 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 **Remarks** field (SQL name: Comment).

2 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.

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, etc.), AssetCenter helps you by only showing those fields that can be used.

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.

In the list of fields and links of a table, unfold the branch marked **Calculated fields**. This branch shows the list of available calculated fields. AssetCenter displays the calculated fields using the format defined via the **Tools/ Options** menu, **Display** tab:

You can add these to the list in the same way as for any other field.

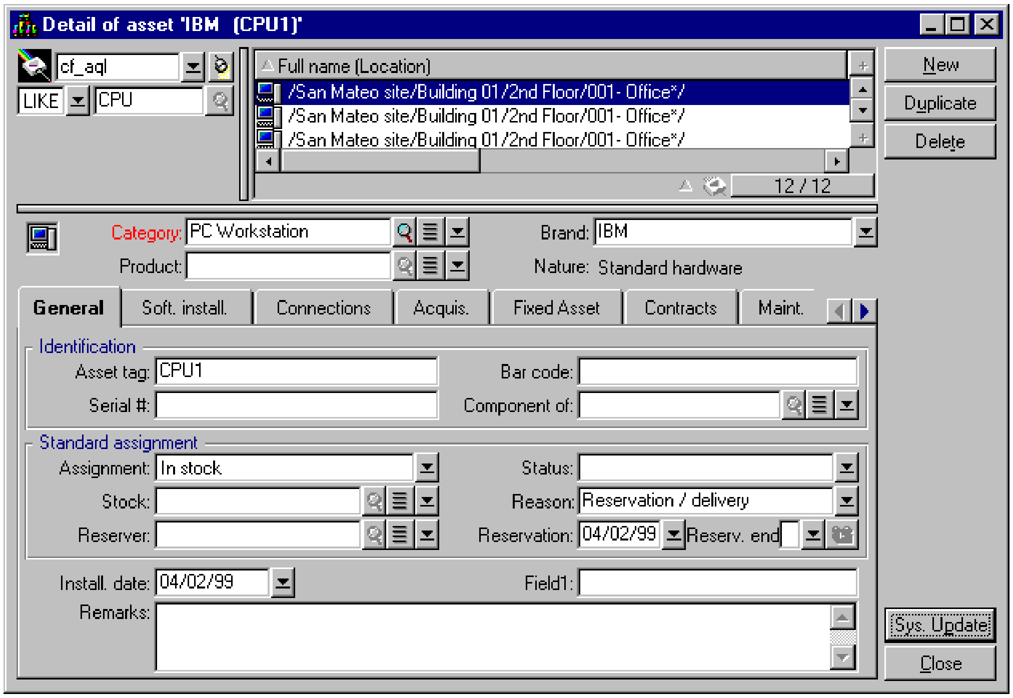
Filtering the records of a table

AssetCenter 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. AssetCenter 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:

Figure 21.2. Example of a calculated field



Detail of asset 'IBM (CPU1)'

cf_aql

LIKE CPU

Full name (Location)

/San Mateo site/Building 01/2nd Floor/001- Office*/
/San Mateo site/Building 01/2nd Floor/001- Office*/
/San Mateo site/Building 01/2nd Floor/001- Office*/

12 / 12

Category: PC Workstation
Brand: IBM
Product:
Nature: Standard hardware

Identification
Asset tag: CPU1
Bar code:
Serial #:
Component of:

Standard assignment
Assignment: In stock
Status:
Stock:
Reason: Reservation / delivery
Reserver:
Reservation: 04/02/99
Reserv. end

Install. date: 04/02/99
Field1:
Remarks:

Additionally, calculated fields can be used by the different modules or functions of AssetCenter:

- AssetCenter Web
- AssetCenter APIs
- Reports
- Forms

For further information on these modules or areas of functionality, please refer to the corresponding documentation.

22 Introduction to wizards

CHAPTER

Use the Tools/ Actions/ Edit menu item, then select a Wizard type action to access wizards

Definition of a wizard

AssetCenter wizards are designed to help you carry out simple and systematic tasks. They offer graphic, user-friendly, step by step guidance through the different phases necessary to carry out a task. AssetCenter ships with several predefined wizards that make it possible, amongst other things, to:

- Move users and assets from one location to another. You select a user (employee), the corresponding assigned assets and the new location. The wizard takes this information and updates the location of the assets and their user.
- Simply manage stocks. The user selects the assets in stock, an employee and a location. The wizard assigns the assets to the selected employee and location.

- Simply gather information in order to perform any given action.
- Facilitate the entering of records.

Note: In addition to the wizards provided with AssetCenter, you can create your own.

Who are wizards aimed at?

Wizards are very useful, for both ordinary users and power users:

- Ordinary users can perform complex tasks without necessarily having to understand the more complex mechanisms of AssetCenter.
- Power users can create their own wizards or customize existing wizards in order to adapt them to the needs of the organization.

Classification of wizards

Wizards can be divided up into two main groups:

- Wizards that exchange information with the AssetCenter database
- Independent wizards

Wizards that exchange information with the AssetCenter database

There are two types of these:

Data-entry wizards

These wizards are intended to facilitate data entry and the creation of records; They populate screens for you. For example, the "New user" wizard guides the user through the creation of a record in the table of departments and employees. Information concerning the employee is gathered by the wizard that in turn creates the corresponding record.

The user does not have to directly enter information in the detail screen. The wizard takes care of this.

Data gathering wizards

Certain actions require a value from the AssetCenter database or the value of a variable. These wizards help to gather and display the necessary information. For example, the "Move" wizard gathers information on the assets to be moved, the user of these assets and their new location. This information is then used to modify the data in the AssetCenter database accordingly.

Independent wizards

Independent wizards are used essentially to perform calculations and display information. You can, for example, create a "Sum" wizard that gives the sum of two values entered by the user.

23 CHAPTER Creating wizards

AssetCenter 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.

Conventions used

The following notation is used to describe the structure of wizards:

Table 23.1. 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".
< >	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 code, the apostrophe designates a line of comments that is not interpreted by AssetCenter.

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, etc.

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: 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, etc.) in a page.
- A variable.
- Etc.

Parent object and child object

If an object "A" contains an object "B":

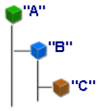
- Object "A" is called the parent object of object "B".
- Object "B" is called the "child object" of object "A".

Warning: 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:

Figure 23.1. 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 AssetCenter 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 AssetCenter.
- 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 AssetCenter.

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 by fulfilled in order to trigger the transition. When the user clicks Next >, the first valid transition is executed (i.e. that for which the conditions are met). If no transitions are valid, the Next > button is disabled.

Structural model

An AssetCenter 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.
- "LONG" (number) or "STRING" (character string) type properties can be used.
- The value of a property is made explicit either via a constant or via a Basic script.

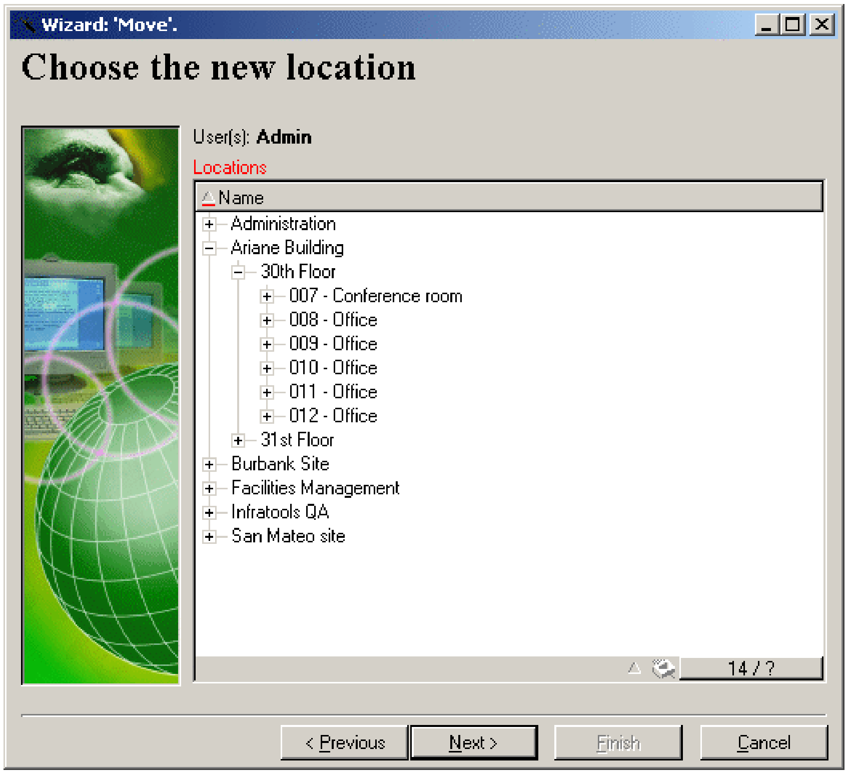
Wizards are made up of pages linked by transitions. The branching from one page to another is determined by the information entered the choices made by the user. The following diagram illustrates the structure of a wizard:

Figure 23.2. Structure of a wizard

Model of a wizard page

A wizard page is organized as follows:

Figure 23.3. Page of a wizard



Choose the new location

User(s): **Admin**

Locations

Name
- Administration
- Ariane Building
  - 30th Floor
    - 007 - Conference room
    - 008 - Office
    - 009 - Office
    - 010 - Office
    - 011 - Office
    - 012 - Office
  - 31st Floor
- Burbank Site
- Facilities Management
- Infratools QA
- San Mateo site

14 / ?

General points

The code of a wizard (entered in the **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", etc.) 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, AssetCenter 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, please 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: Warning: The space between the ("{") brace and the node type must be respected. If this is not the case, AssetCenter will refuse to execute 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).

Declarative model

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 "O". "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

3A 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 is as follows:

```
{node.node.node[.property][.method()]}
```

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()
```

The following table list the properties having methods:

Table 23.2. List of methods that can be used

Property concerned	Method	Description
DBLISTBOX.TABLE	LABEL()	Returns the label of the table allowing you to generate the DBLISTBOX control.
DBEDIT.TABLE	LABEL()	Returns the label of the table allowing you to generate the DBEDIT control.
DBEDIT.FIELD	LABEL()	Returns the label of the field of the table allowing you to generate the DBEDIT control.
LISTBOX.VALUE	COUNT()	Returns the number of lines in the LISTBOX control.
LISTBOX.VALUE	CELL(i,j)	Returns the contents of the cell (i,j) of the table type property.
LISTBOX.VALUE	VALUE(i)	Returns the value of line i of the table type property.
LISTBOX.VALUE	LINE(i)	Returns the contents of line i of the table type property.
LINKEDIT.TABLE	LABEL()	Returns the label of the table allowing you to generate the LINKEDIT control.

Table type property

Table type properties are properties whose value is defined according to the following format:

```
<Colonne|Colonne|Colonne|...>=<Value of the line>,
 <Colonne|Colonne|Colonne|...>=<Value of the line>,
 ...
```

Note: The <Value if the line> is equal to the identifier ("Id") of the corresponding record.

The values of these properties can be viewed in the form of a table:

Table 23.3. "Table" type property

		Column 1	Column 2	Column 3
Line number: 1	**Value of the line (e.g.: 18)**	Cell (1,1)	Cell (2,1)	Cell (3,1)
Line number: 2	**Value of the line (e.g.: 29)**	Cell (1,2)	Cell (2,2)	Cell (3,2)
Line number: 3	**Value of the line (e.g.: 78)**	Cell (1,3)	Cell (2,3)	Cell (3,3)
Etc.	Etc.	Etc.	Etc.	Etc.

Example

Let's consider the property "VALUES" having as its value the result of a query on the table of departments and employees. 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|Gérard=32,Lubeck|Alexandre=64,Daquin|William=24"
```

This value can be viewed in the form of a table:

Table 23.4. Example

		Last name	First name
1	**32**	Colombo	Gerard
2	**64**	Lübeck	Alexander
3	**24**	Daquin	William

Using the global variables CTXTTABLE and CTXTSELECTION

The contents of these variables can be recovered using the following syntax:

```
[CTXTTABLE]
[CTXTSELECTION]
```

The following table presents the features of these two variables:

Table 23.5. Global variables features

Name of the variable	Description of the variable	Comment
CTXTTABLE	Contains the active table at the time the wizard is launched. "String" type variable.	This variable is automatically populated by AssetCenter. The user cannot force the value.
CTXTSELECTION	Contains the list of internal identifiers of records selected at the time the wizard is launched. "String" type variable.	This variable is automatically populated by AssetCenter. The user cannot force the value.

Sequencing wizards

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.

Execution

In order for a wizard A to trigger a wizard B, its Finish not must 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".

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 takes precedence over that in wizard B.

Basic functions

In addition to the generic functions of AssetCenter (with the exception of the "AmCounter" function), wizards accept the following additional functions:

- AmComputeString()
- AmProgress()
- AmLog()
- AmMsgBox()
- AmExecTransition()

Attention: When you call Basic functions from a wizard script, you must always assign the value returned by the function to a variable. Otherwise, the Basic will return an error. Thus, 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})
```

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 table lists all the general properties that can be defined in a "Root" node:

Table 23.6. 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. "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. "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, it conserves (storing them in the "ini" file) the values previously entered for the next time it is executed. "Boolean" type property.	{GLOBAL=Dim Filter As String}	

Name of the property=Value	Description of the property	Example	Comment
SERIALIZE=<TRUE|FALSE>	Allows you to serialize the wizard (=TRUE) or not (=FALSE). If the wizard is serialized, it conserves (storing them in the "ini" file) the values previously entered for the next time it is executed. "Boolean" type property.	SERIALIZE=TRUE	By default, this property is set to "FALSE".

Table 23.7. 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>"	Defines the bitmap type graphical file (.bmp) to be displayed in the wizard. "String" type property.	IMAGE="C:\Images\Page1.bmp"	If no value is defined for this property, then no image will be displayed. The path of the graphics file depends on the AssetCenter Config folder. AssetCenter first searches for the imagine in the database If you define a value for "IMAGE16", this property is used instead of "IMAGE" when the color depth of the screen is 16.
WIDTH=<Width>	Defines the default width ("<Width>") of the window of the wizard. This is expressed in twips. "Long" type property.	WIDTH=6000	
HEIGHT=<Height>	Defines the default height ("<Height>") of the window of the wizard. This is expressed in twips. "Long" type property.	HEIGHT=5000	

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".

Table 23.8. Sub-nodes of "Root" node

Node type	Description
PAGE <Name of the page>	Describes a page of the wizard.
TRANSITION <Name of the transition>	Describes a transition between two pages.
FINISH <Name of the node>	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 <Name of the node>	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).

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 table lists all the properties that can be defined in a "Page" node:

Table 23.9. 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. "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 with the page is accessed. "Script" type property.	{ONENTER = AmMsgBox ("Hello") }	

Table 23.10. Physical properties of a "Page" node

Name of the property=Value	Description of the property	Example	Comment
IMAGE="<Path of the bitmap file>" IMAGE16="<Path of the bitmap file>"	Defines the bitmap type graphical file (.bmp) to be displayed in the page of the wizard. "Script" type property.	IMAGE =" C:\Images\Page1.bmp"	If no value is defined for this property, it inherits the value of the "IMAGE" property of the "Root" node. If you define an empty value for this property, no image is displayed. 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:

Table 23.11. 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

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:

Table 23.12. 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 original page of the transition. "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 target page of the transition. "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. "Script" type property that returns a Boolean.	CONDITION= {Comment}="user"	This property is not available in "Start" nodes.
DO=<Script>	Defines a script to be executed at the time of the transition. "Script" 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.

Definition of a 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.

Table 23.13. Logical property 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 wizard to be executed at the end of the current wizard. "String" type property	CHAIN = "Move"	

Note: The PARAMS node allows you to pass parameters to the following wizard.

Definition of a 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.

Definition of 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.

Table 23.14. 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. "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.

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. AssetCenter 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 controls have properties in common. However, there are properties specific to certain controls. This section contains information on the following:

- Common properties
- The CheckBox control
- The ComboBox control
- The ListBox control
- The Label control
- The ProgressBar control
- The CommandButton control
- The OptionButtons control
- The DBListBox control
- The DBQueryBox control
- The DBEdit control
- The DBTable control
- The DBPath control
- The LinkEdit control
- The TextBox control

Common properties

The following table lists optional properties applicable to all controls:

Table 23.15. Logical properties common to all controls

Name of the property=Value	Description of the property	Example	Comment
MANDATORY=<TRUE|FALSE>	Forces the user to populate the control in order to validate a transition. "Boolean" type property.	MANDATORY=TRUE	This property is not available for "CHECKBOX" and "LABEL" controls
VALUE=<Value>	Defines the default value of the control when created. <Value> depends on the control concerned. "Boolean" or "String" type property.	For example, if it is a "CHECKBOX" control, <Value> can be "TRUE" or "FALSE".	

Table 23.16. Physical properties common to all controls

Name of the property=Value	Description of the property	Example	Comment
VISIBLE=<TRUE|FALSE>	Defines if the control is visible (=TRUE) or not (=FALSE). "Boolean" type property.	Label1.Visible=TRUE	
ENABLED = <TRUE|FALSE>	Defines if the control is active (=TRUE) or not (=FALSE). "Boolean" type property. "Boolean" type property.	Choice1.Enabled=FALSE	
READONLY = <TRUE|FALSE>	Defines if the value of the control is read-only (=TRUE) and therefore cannot be modified by the user; or editable (=FALSE)	READONLY=TRUE	
LABEL = "<Text of the label>"	Defines an optional text, displayed above the control. "String" type property.	Choice1.Label="Select person"	This label supports HTML

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:

Table 23.17. Property of "CHECKBOX" control

Name of the property=Value	Description of the property	Example
TEXT="<Text>"	Defines the text of the check box. "String "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:

Table 23.18. Physical properties of the "COMBOBOX" control

Name of the property=Value	Description of the property	Example	Comment
VALUES="<Name=Value, Name=Value, Name=Value,...>"	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. "String" type property.	VALUES="Table of assets=asset, User=user"	Example VALUES = "A,B,C" is equivalent to VALUES = "A=1,B=2,C=3" If you omit the "Value", AssetCenter will automatically assign one.

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:

Table 23.19. 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. "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-selection (=TRUE) or not (=FALSE). "Boolean" type property.	MULTISEL=TRUE	
COLTITLE = "<Column|Column|Column>"	Defines the title and properties of columns in the list. "Column" defines the text of the column. "String" type property.	COLTITLE = "Name|FirstName"	
COLWIDTH = "<Width|Width|Width...>"	Defines the size of the column proportionally to the overall size of the control. "String" type property.	COLWIDTH = "50|50"	

Name of the property=Value	Description of the property	Example	Comment
VALUES = "<Texte|Texte|...= Value, Text|Texe|...= Value,...>"	Defines value couples ("Text|Text|..."= "Value") for the "LISTBOX" control. "Text|Text|.." defines the text to 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. "String" type property.	VALUES="Table of assets=asset, , User=user,"	Example VALUES="A,B,C" is equivalent to VALUES="A=1,B=2,C=3" If you omit the "Value", AssetCenter will automatically assign one. This property can be populated directly or using the AmdbGetList, function by writing, for example: VALUES = AmDbGetList ("SELECT Name, FirstName FROM amEmplDept WHERE Name Like 'A%'", "|", ",", "=") Do not confuse the "VALUES" and "VALUE" properties.

Table 23.20. 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. "String" type property.	TABLE=**amAsset**
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 AssetCenter to populate the field. AssetCenter will first take the values of the "COLTITLE" property (if this exists) into account to define the titles of columns. "String" type property.	COLNAME="Name|FirstName"
EDITABLE = <"booléen|...|booléen">	Specifies if the list is editable for each column. "String" type property.	EDITABLE="1|0|1|1"

The Label control

The "LABEL" control simply defines a label. This control has the following property:

Table 23.21. 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 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:

Table 23.22. Physical properties of the "OPTIONBUTTONS" control

Name of the property=Value	Description of the property	Example
VALUES="<Title=Value, Title=Value, Title=Value,....>"	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. "String" type property.	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) "Boolean" type property.	BORDER= TRUE

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:

Table 23.23. 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. The "VALUE" property indicates the current value of the control. "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:

Table 23.24. 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. "Long" type property.	WIDTH=250
HEIGHT=<Height>	Defines in twips the height of the button. "Long" type property.	HEIGHT=125
CAPTION=<Text>	Defines the text displayed within the button. "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 AssetCenter 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:

Table 23.25. 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. "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). "String" type property.	COLNAME = "Name|FirstName"	
COLWIDTH = "<Width|Width|Width|...>"	Defines the display format for the data extracted from the database, as a percentage of the overall size of the "DBLISTBOX" control. "String" type property.	COLWIDTH="40|60"	
LISTHEIGHT = <Percentage>	Defines the size of the "DBLISTBOX" control in relation to other "DBLISTBOX" controls in the wizard as a whole. "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). "Boolean" type property.	TREE=TRUE	By default, this property is set to "FALSE"

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). "Boolean" type property.	MULTISEL=TRUE	
DBLCLICK = <TRUE|FALSE>	If this property is set to TRUE, AssetCenter will simulate clicking the **Next** button of the current page.	DBLCLICK=FALSE	
FILTER= "<Condition>"	Defines the AQL "WHERE" condition to filter records to be processed in the query. "String" type property.	FILTER= "User.lEmplDeptId='Colombo, Gerard' "	

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 AssetCenter database.

Properties

In addition to the optional properties common to all controls, the "DBQUERYBOX" control recognizes the following properties:

Table 23.26. 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. "String" type property.	QUERY="SELECT Name, FirstName FROM amEmplDept WHERE Location=Ariane Building'"	
COLTITLE="<Column|Column|...>"	Defines the title of columns in the list. "String" type property.	COLTITLE= "Name|FirstName"	
COLWIDTH="<Width|Width|...>"	Defines the size of columns in the list as a percentage of the overall size of the control. "String" type property.	COLWIDTH= "50|50"	
LISTHEIGHT=<Percentage>	Defines the size of the "DBQUERYBOX" control proportionally to other "DBQUERYBOX" control in the wizard as a whole. "Long" type property.	LISTHEIGHT=50	If there are two "DBQUERYBOX" 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). "Boolean" type property.	TREE=TRUE	By default, this property is set to "FALSE"

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). "Boolean" type property.	MULTISEL=TRUE	
DBLCLICK=<TRUE|FALSE>	If this property is set to TRUE, AssetCenter will simulate clicking the **Next** button of the current page. "Boolean" type property.	DBLCLICK=FALSE	

The DBEdit control

The "DBEDIT" control creates a control identical to that used to populate a field in the AssetCenter database. The control differs according to each field type (date, monetary, etc.).

Note: The magnifier button in this control allows you to select values that are effectively in the database, but you can enter another value.

Properties

In addition to the optional properties common to all controls, the "DBEDIT" control recognizes the following properties:

Table 23.27. Mandatory physical properties of the "DBEDIT" control in "Normal" mode

Name of the property=Value	Description of the property	Example / Comment
TABLE=<SQL name of the table>	Name of the table containing the field whose control you want to copy. "String" type property.	TABLE=**amAsset**
FIELD=<SQL name of the field>	Name of the field whose control you want to copy. "String" type property.	FIELD=**seAcquMethod**

The DBTable control

The "DBTABLE" control creates a control for entering a table in the AssetCenter database.

Properties

This control does not have any additional properties.

The DBPath control

The "DBPATH" control creates a control for entering a field in the AssetCenter database.

Properties

In addition to the optional properties common to all controls, the "DBPATH" control must have the following mandatory property:

Table 23.28. Mandatory logical property of the "DBPATH" 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 field. "String" type property.	TABLE=**amAsset**

The LinkEdit control

The "LINKEDIT" control creates a control for entering a link in the AssetCenter database.

Properties

In addition to the optional properties common to all controls, the "LINKEDIT" control has the following properties:

Table 23.29. Mandatory 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. "String" type property.	TABLE=**amAsset**
FILTER=<WHERE clause of an AQL query>	Defines a filter using the result of an AQL query. "String" type property.	This property is optional.

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:

Table 23.30. 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

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 o 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 an to consider this section as a guide in case of problem.

1. Example of creating a wizard
2. Step #1- Analyze your needs
3. Step #2- Define how the wizard is organized

Step #1: Analyze your needs

The objective of this wizard is move assets from one location to another. For this we need:

1. To identify the assets to be moved.
2. Choose the new location for these assets.

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 #2: Define how the wizard is organized

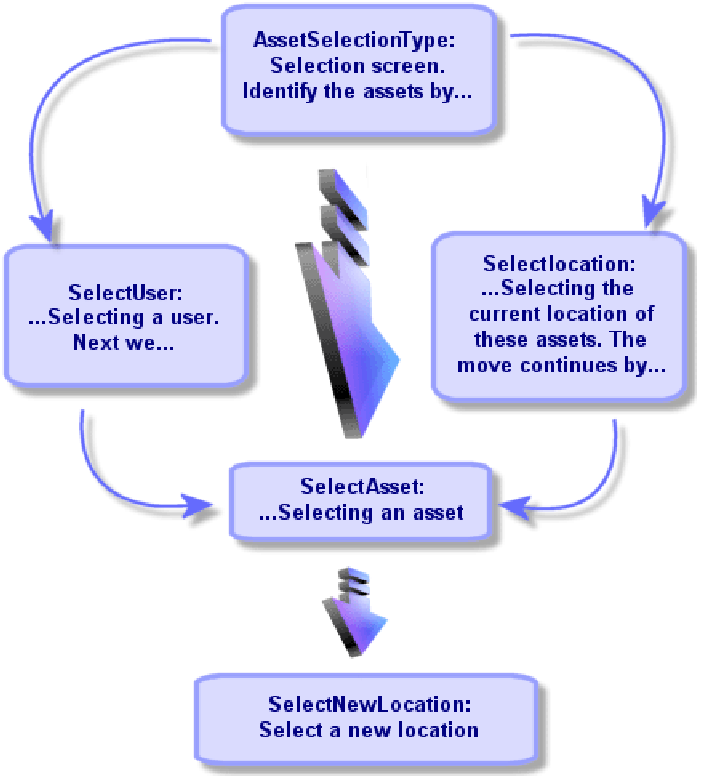
According to the needs defined in step 1, we need to define the organization of the wizard. I.e.:

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:

Figure 23.4. Wizard organization



Using this flow-chart, we can now define the transitions page by page:

Table 23.31. Definition of transitions

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:

Table 23.32. Page contents

Page	Objective of this page?	Control to use
AssetSelectionType	Allows the user to choose between three possibilities	A "CHOICE" control
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°#3: Transcribe the structure of the wizard using the scripting language

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.

```
================================
;(c) Peregrine Systems 1999
;=================================================

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 lEmplDeptId IN (" &
 {pgUser.Users} & ")" )
  }
  { LABEL LABEL1
    CAPTION = "User(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)
    lErr = AmUpdateRecord(hRecord)
    lErr = AmReleaseHandle(hRecord)
   End If
  Next i

  lErr = amCommit()

  RetVal = 0
  Exit Function


  ErrHandler:
   On Error Goto 0
   AmLog(AmLastError() & " - " &
AmLastErrorMsg())
   AmLog("The transaction has been cancelled")

   RetVal = 1
   Exit function
 }
 SUMMARY = 1
}
```

Using the graphical editor

AssetCenter 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, the understanding of which is essential in order 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 by a "Wizard" type action (SQL name: seActionType).

Overview of the interface

To access the graphical editor, select the **Tools/ Actions/ Edit** 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

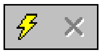
Four edit commands are available:

- ☒ 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.

Execute and debug commands

These commands allow you to compile, debug and execute the script:

Figure 23.5. 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, AssetCenter 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, AssetCenter 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:

AssetCenter 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, AssetCenter 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, etc.), AssetCenter 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), AssetCenter 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 be 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.

Frequently asked questions

This chapter aims to answer the questions you are likely to ask when creating a wizard.

Question

The following code does not work:

```
{lbxMyListBox.Values.Count}
```

Answer

You must enter opening and closing parentheses in the syntax of the method. Here is the corrected code:

```
{lbxMyListBox.Values.Count()}
```

Question

The following code does not work:

```
{lbxMyListBox.Line(lRow)}
```

Answer

The "LINE" method is associated with the "VALUES" property of the "LISTBOX" control. Here is the corrected code:

```
{lbxMyListBox.Values.Line(lRow)}
```

Question

The following code does not work:

```
{lbxMyListBox.Values.Line({lbxTmp})}
```

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)}
```

Question

The following code, which assigns a fixed value to a property, does not work:

```
{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("Proprerty", 123)
```

Note: Don't forget to recover the return code ("irc" in this example), even if you are don't need to use it.

Question

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 if 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 AssetCenter administrator)
2. Executes the script contained in the "FINISH.DO" node (write-access authorized according to the user's access rights)

Question

Error messages that appear when executing wizard are sometimes incomplete.

Answer

Press SHIFT+F9 to display the debugger. Error messages in the history window are often more explicit.

Question

When the "DBLISTBOX" control is used in a wizard page, performance is impacted. Is this normal?

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.

Question

How do I 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 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"
```

Question

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("aam", "AssetCenter",
"OpenTable(amAsset)")
```

Question

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.

24 Tuning AssetCenter for use in a WAN environment

CHAPTER

WAN networks are often characterized by:

- Low bandwidth.
- High latency.

It is possible to configure AssetCenter in order to minimize these disadvantages. However, these configurations are made to the detriment of certain functions of AssetCenter.

This chapter provides some tips that can be used in order to accommodate the limitations of WAN networks. All the same, it is important to carry out tests to measure the trade-off between the faster reaction times and the loss of functionality.

Options in the Tools/ Options menu item

You can limit the length of accesses made to the database using the following options:

- **Type-ahead after** option in the **Navigation** tab: You can either deactivate Type-ahead, or attribute a high value (for example, Type-ahead after 10 000 ms).
- **Trees in drop-down lists** option in the **Navigation** tab: You can deactivate this option, since tree-views are more costly in terms of performance that list views.

  However, you lose the user-friendly nature of tree-views in drop-down lists.

You can limit the exchange of information between the client machine and the database server by adjusting the following options:

- **Do not load for longer than** and **Do not load more than** options in the **Lists** tab (main lists or others): We recommend limiting the number of lines loaded (for example, you can specify a maximum value of 50 lines in main lists and 15 in tab lists). It is up to you to determine the number of lines to be loaded, according to any filters you apply to the lists being displayed, and the likelihood of finding the required information in a given number of lines.
- **Test new messages**, **Messaging** tab: You can configure this option in order to test for messages on connecting to the database only, or space out the verification intervals (every 10 minutes, for example).
- **Caches** tab: You can increase the length of intervals used to define the frequency of refreshing the caches (**Every** column), or even disable the refreshing of caches during a session. In this case, caches are loaded on connecting to the database only.

  It is likely that if you do not refresh the caches on a regular basis, the data being displayed will not be up to date. However, the majority of cached data items are generated when AssetCenter is installed and are not subject to regular variations (itemized lists, dictionary of features, currencies, business calendars, etc).

Lists

Setting the parameters for a particular list

Main list and tab lists can be configured via the **Configure list** menu item in several cases:

- Lists displayed via the menus allowing you to access tables (**Portfolio/ Assets and batches** menu item, for example).
- Lists displayed by views (**Tools/ Views** menu item).
- Selection lists (**Select link** shortcut menu item).
- Lists that appear in certain tabs of details.

Sorting lists

These lists can be sorted in several ways:

- by selecting your own sort conditions (**Sort** column),
- by using predefined indexes (**Sort by index** field).

The resulting performance of these two options sometimes differ. It is not possible to predict which option offers the best solution.

You need to try both solutions for each list in AssetCenter before deciding on which method is the best for your database.

Filters

Lists can also be filtered.

The time it takes to display a list increases with:

- The number of filtering criteria.
- The distance of the tables to which the filter criteria applies (distance from the table whose list displays the contents).
- The number of **OR** clauses in the filter query.

Selecting the columns to display

The time it takes to display a list increases with:

- The number of columns to display.

- The distance of the tables containing the fields and links are to be displayed.

Displaying in table or tree view.

The **tree** view takes longer to display than the **table** view.

Displaying icons in lists

Icons take longer to appear than does the text.

Setting list parameters at the database level

Certain interface options can affect the time it takes to display a list.

To access these options:

1. Select the **Options/ Tools** menu item.
2. If necessary, modify the following options:
   - **Lists/ Other lists/ Do not load for longer than**
   - **Lists/ Other lists/ Do not load more than**
   - **Lists/ Main lists/ Do not load for longer than**
   - **Lists/ Main lists/ Do not load more than**

By reducing the number of lines loaded in a list, you can decrease the time it takes to display a list.

By reducing the maximum amount of time allotted for displaying a list, you thus limit the number of lines displayed according to what you deem a reasonable amount of loading time.

Note: These options are stored in the AssetCenter database and are the same for all users who access it.

To learn more about these interface options, refer to the **Introduction** guide, chapter **Customizing a client workstation**, paragraph, **General AssetCenter interface options**.

Setting list parameters at the AssetCenter client level

The **ArrayFetchingSize** parameter is used by all the DBMSs supported by AssetCenterfor an AssetCenter data connection.

To learn about this parameter's utility, you need to know that a DBMS sends the requested records in packets from an AssetCenter client. The size of these packets (expressed by number of records) is defined by the **ArrayFetchingSize** parameter.

This parameter is defined at the level of each AssetCenter client in the **amdb.ini** file and for each AssetCenter connection. This file is located at the root of the Windows installation folder.

If the **ArrayFetchingSize** parameter is not in the **amdb.ini** file, then its default value is **30**.

This parameter is integrated with the following parameters of the **Tools/ Options** menu:

- **Lists/ Other lists/ Do not load more than**
- **Lists/ Main lists/ Do not load more than**

Non-optimized example for WAN

- Let's suppose that the **Do not load more than** option is set to **20**.
- Let's also suppose that **ArrayFetchingSize** is set to **30**.
- Then AssetCenter will fetch the list in 7 rounds (since `200 / 30 = 6,7`). This requires more time than if the list was fetched just once.

Tip: This amount of time is generally insignificant for a LAN but can be quite noticeable for a WAN.

Optimized example for a WAN

If it takes a long time to display a list, you need to configure the application to fetch records just once.

Here's the rule you need to apply:

```
ArrayFetchingSize = Do not load more than + 1
```

Note: We tested this scenario with a WAN using a 250 ms ping. By optimizing this parameter, we gained 1.5 s upon displaying a list of 200 records.

Example 24.1. Example of application:

- **Do not load more than** equals **200**
- Set the value of **ArrayFetchingSize** to **210**
- Thus, AssetCenter fetches just once.

To modify amdb.ini

1. Edit the **amdb.ini** file.
2. Search for the selection `[<AssetCenter connection name to optimize>]`
3. Find out if a line already exists in the section that starts with `ArrayFetchingSize=`.

   If this is the case, modify the value of the existing parameter.
4. If this is not the case, add a complete line in the section: `ArrayFetchingSize=<Parameter value>`

Tip: This must be done on each client workstation.

Stripping down screens

To increase the reaction times of the application, you can restrict the number of data items appearing on screen, by only displaying those columns, lists and tabs that are strictly necessary.

Connection cache

You can also activate the connection cache, via the **Cache** tab in the connection detail screen:

Activating the connection cache:

- Reduces connection time to the database.
- Also saves time if you use images and icons.

In general, the default cache sizes are well optimized.

Access restrictions

The displaying of detail and list windows is slowed down when there are access restrictions for the given login used. This is due to the fact that AssetCenter performs a test before displaying the data.

If you have a doubt, display the list or detail with an unrestricted login and compare the display speed.

Delete any optional access restrictions as you see fit.

Applying the configuration of one client to other clients

Once you have optimized the performance of a given workstation, you now need to propagate the configuration modifications to the other client workstations.

In order to save time, you can copy the ***.ini** files corresponding to the modifications you have made.

For details of the list, contents and location of the ***.ini** files, refer to the **Administration** guide, chapter ***.ini files**.

25 Integrating external applications

CHAPTER

This chapter presents the integration of external applications with AssetCenter.

InfraTools Remote Control

Available functions

The integration of InfraTools Remote Control enables you to take control of a remote computer registered in your **Computers** table (amComputer) directly from AssetCenter.

Prerequisites

In order for these functions to be available, you must have met the following requirements:

- Your AssetCenter license gives you access to the InfraTools Desktop Discovery integration.

- An InfraTools Remote Control manager is installed on the machine that triggers the remote control session.
- An InfraTools Remote Control agent is installed on the computer to be controlled.
- The **Remote Control** action stored in the AssetCenter database is correctly configured:

  1. Execute AssetCenter
  2. Select the **Tools/ Actions/ Edit** menu item.
  3. Select **Remote Control** from the list.
  4. Select the **Executable or DDE** tab.
  5. Modify the **Parameters** field if necessary.

     By default, the parameter line is:

     ```
     -host:[Name] -mode:gui -close
     ```

     To learn more about the parameters, refer to the InfraTools Remote Control **User's guide**, chapter **Using the manager**, section **Using the manager from the command line**.

     Add the **-type:srv** parameter if the computers to control are part of an InfraTools Remote Control server (and not **Direct Access** computers).
- The computer to control is in the AssetCenter **Computers** table (amComputer).

  The **Name** field in the AssetCenter database and the name of the computer in the InfraTools Remote Control database have the same value.
- An InfraTools Remote Control agent is installed and activated on the computer to control.

Taking control of a computer from AssetCenter

To take control of a computer:

1. Select this computer in the **Computers** table (amComputer).
2. Perform one of the following actions:

- Click **Control** in the window of the computer's detail.
- Click in the toolbar.

Tip: If this icon is not in the toolbar, add it by:

1. Selecting the **Tools/ Customize toolbar** menu item.
2. Selecting the **Tools** tab.
3. Selecting the **Tools** category.
4. Dragging the icon to the toolbar.

- Right-click and select **Remote control** from the shortcut menu that appears.

For more information on the integration of InfraTools Remote Control, refer to the InfraTools Remote Control **User's guide**, chapter **Integrating your other applications with InfraTools Remote Control**, section **Integrating InfraTools Remote Control with AssetCenter**.

InfraTools Desktop Discovery

Available functions

The integration of InfraTools Desktop Discovery with AssetCenter enables you to perform the following operations:

- Trigger the scan of a computer in the **Computers** table (amComputer) from AssetCenter
- View the results of the scan.

Tip: AssetCenter also enables you to perform automatic scans on multiple computers. This function is described in chapter

Managing deadlines with AssetCenter Server, section Configuring the modules monitored by AssetCenter Server, sub-sections:

- **Send the scanner to the computers module** (SendScan)
- **Get scanner results** module (GetFsf)
- **Update the database using data from the scanners** module (IddAc)

Prerequisites

In order for these functions to be available, you must have met the following requirements:

- Your AssetCenter license gives you access to the InfraTools Desktop Discovery integration.
- An InfraTools Remote Control manager is installed on the machine that triggers the scan.

  This manager is configured to be integrated with InfraTools Desktop Discovery:

  1. Execute the InfraTools Remote Control manager.
  2. Select the **Edit/ Options** menu item.
  3. Activate the **IDD integration** option.
  4. Populate the **Scanner command line** to use for the scanner option: To find out which commands to use in the scanner, refer to the **scanner.hlp** help file, located in the **bin32\scanners** subfolder of the Connect-It installation folder.

     Note: This file is only available if you have installed the Automatic inventory package from the AssetCenter CD-ROM.

  5. Populate the **Check if scan finished every** option.

6. Populate the **FSF file download location** option: This is the folder of the machine that launches the scan where you want to store **.fsf** files of the scanned computers.
7. Populate the **Folder upload location** option: This is the folder of the scanned computer where you want to upload the scanner.
8. Populate the **FSF file to download** option: This is the path of the **.fsf** file on the scanned computer. It must correspond to the file indicated when you configured the scanner that you are using. For the **scan32.exe** installed with Connect-It, this file is named **default.fsf**.
9. Populate the **Scanner to upload to remote computer** option: This is the path of to the scanner to upload. You can use any scanner you want, however, a default scanner is already provided with Connect-It: **scan32.exe.** It is located in the **bin32** sub-folder of the Connect-It installation folder.
10. Populate the **Start the viewer once the FSF file is fully downloaded** option: If you select **Yes**, the InfraTools Desktop Discovery viewer appears automatically after the scan.

- An InfraTools Remote Control agent is installed on the computer to be scanned.
- The **Remote Scanner** action stored in the AssetCenter database is correctly configured:

  1. Execute AssetCenter
  2. Select the **Tools/ Actions/ Edit** menu item.
  3. Select **Remote Scanner** from the list.
  4. Select the **Executable or DDE** tab.
  5. Modify the **Parameters** field if necessary.

     By default, the parameter line is:

     ```
     -host:[Name] -mode:scan -viewscan -close
     ```

     To learn more about the parameters, refer to the InfraTools Remote Control **User's guide**, chapter **Using the manager**, section **Using the manager from the command line**.

Add the **-type:srv** parameter if the computers to scan are part of the InfraTools Remote Control servers (and not **Direct Access** computers).

- The computer to scan is in the AssetCenter **Computers** table (amComputer).

  The **Name** field in the AssetCenter database and the name of the computer in the InfraTools Remote Control database have the same value.
- An InfraTools Remote Control agent is installed and activated on the computer to be scanned.

Scanning a computer from AssetCenter

To scan a computer from AssetCenter:

1 Select this computer in the **Computers** table (amComputer).

2 Perform one of the following actions:

  - Click **Scan** in the window of the computer's detail.
  - Click in the toolbar.

Tip: If this icon is not in the toolbar, add it by:

1 Selecting the **Tools/ Customize toolbar** menu item.

2 Selecting the **Tools** tab.

3 Selecting the **Tools** category.

4 Dragging the icon to the toolbar.

  - Right-click and select **Launch a remote scanner** in the shortcut menu that appears.

Note: Integrate InfraTools Desktop Discovery with AssetCenter means integrating InfraTools Desktop Discovery with InfraTools Remote Control. For further information, refer to the InfraTools Remote Control manual entitled User's Guide, chapter Integrating your other applications with InfraTools Remote Control, section Integrating InfraTools Desktop Discovery with InfraTools Remote Control.

Installing a full version of InfraTools Desktop Discovery

AssetCenter is only provided with certain InfraTools Desktop Discovery components.

Here are some good reasons for why you should acquire the full version:

- You can generate some other scanners, not provided with AssetCenter's **Automatic inventory** package (**.exe** files).
- You can add new software signatures, which can be recognized by the scanners (**.sai** files)

Knowlix

The integration of Knowlix with AssetCenter enables you to search the Knowlix knowledge base.

Once you have installed the version 4.6 or later of Knowlix Frontline on your computer, the integration with AssetCenter is completely automatic: Every time that you select the **Search in Knowlix Frontline** command in AssetCenter, a DDE script automatically launches Knowlix or puts in the foreground.

To obtain information from the Knowlix knowledge base:

1. Highlight the text of the field on which you want to obtain information.
2. Perform one of the following actions:

- Click 🔍

Note: This icon must be added to the toolbar, if you haven't done it already, using the Tools/ Customize toolbar menu item.

- Select **Search in Knowlix** from the **Tools** menu.

3 Wait for Knowlix to appear.

4 Select one of the solutions proposed by Knowlix.

5 To recover this solution from the Clipboard, perform one of the following actions in AssetCenter:

- Click 🔍

Note: This icon must be added to the toolbar, if you haven't done it already, using the Tools/ Customize toolbar menu item.

- Select **Recover solution from Knowlix** from the **Tools** menu.

AutoCAD integration

Presentation of AutoCAD/AssetCenter integration

You have the possibility of dynamically linking and tracking your AssetCenter data with the data in AutoCAD and vise versa.

The migration and tracking of your data is done using a Connect-It scenario.

The AutoCAD functionality of AssetCenter adds two new menus in AutoCAD that enable you to import and view locations, employees and assets attached from AssetCenter.

Installation

Creating an AssetCenter database compatible with AutoCAD

Note: An AssetCenter demonstration database that supports AutoCAD is created during the AssetCenter installation. This database is located in the Peregrine/AssetCenter/acadi/db folder.

AssetCenter Database Administrator enables you to create an AssetCenter database that supports AutoCAD integration.

To create a database that supports AutoCAD:

- Open a description file (.dbb).
- Choose **Create a database** from the **Action** menu.
- Perform the following steps, described in the "Creating an empty AssetCenter database" section:
- Select the **Use AutoCAD** integration option.
- Click **Create**.

FacilityCenter users

If you already have the AutoCAD integration module provided with FacilityCenter, use the scenario provided with AssetCenter to synchronize the AutoCAD data.

Configuring Connect-It

The Connect-It scenario provided with AssetCenter is located in the **Peregrine/AssetCenter/acadi/scenario** folder.

Configuring connectors

You must configure the AutoCAD connectors and AssetCenter to be able to use the **acadi.scn** scenario.

Configuring the AssetCenter connector

1 Open the **acadi.scn** scenario.

2. In the **Scenario diagram** screen, right-click the Asset Management connector and select **Configure connector** from the shortcut menu that appears.
3. In the wizard that appears, populate the **Name** field and **Description** field, if necessary, then click **Next**.
4. Select your AssetCenter database (**Connection** field) and populate the **Login** and **Password** fields of the ODBC connection before clicking **Next**.
5. Click **Finish**.

Configuring the AutoCAD connector

1. In the **Scenario diagram** screen, right-click the AutoCAD connector and select **Configure connector** from the shortcut menu that appears.
2. In the wizard that appears, populate the **Name** field and **Description** field, if necessary, then click **Next**.
3. Select your AssetCenter database that is compatible with AutoCAD (**Connection** field) and populate the **Login** and **Password** fields before clicking **Next**.
4. Click **Finish**.

Configuring FacilityCenter

Refer to the section Configuring the AssetCenter connector to configure the Asset Management connector. Then, configure the FacilityCenter connector using the information described below:

1. In the **Scenario diagram** screen, right-click the FacilityCenter connector and select **Configure connector** from the shortcut menu that appears.
2. In the wizard that appears, populate the **Name** field and **Description** field, if necessary, then click **Next**.
3. Select your FacilityCenter database (**Connection** field) and populate the **Login** and **Password** fields before clicking **Next**.
4. Click **Finish**.

Using AutoCAD integration

Graphic interface

The AutoCAD module installed by AssetCenter adds two menus in AutoCAD: DrawTools and FacilityCenter.

Overview

You need to have already correctly configured the synchronization of data in Connect-It before starting to work on your AssetCenter applications or on AutoCAD.

Synchronizing data with Connect-It

The **acadi.scn** Connect-It scenario enables you to synchronize and migrate data between AssetCenter and AutoCAD.

This scenario enables you to replicate data between AssetCenter and AutoCAD, which you can do periodically using a certain programming in Connect-It. Example: Data replication between AssetCenter and AutoCAD is performed each Friday. Refer to the Connect-It user's guide to find out more about this scenario.

You must replicate the AssetCenter data in AutoCAD (**Produce** contextual menu item in the AssetCenter connector) in order to be able to use it in AutoCAD.

Using AssetCenter data in AutoCAD

To access your AssetCenter database that is compatible with AutoCAD:

1. Use the **FacilityCenter/Property Portfolio** menu item.
2. Select your AssetCenter database.
3. Choose your location (building, floor, room, etc.) and click **Attach**.
4. AutoCAD tells you when it is connected to the database, and you can starting working on the selected map.

After having worked in AutoCAD and saved your modifications, and before having used AssetCenter, launch the Connect-It scenario in order to synchronize your data (**Produce** contextual menu item in the AutoCAD connector).

Using AutoCAD data in AssetCenter

After having migrated your AutoCAD data to Connect-It, the **Employees**, **Locations** and **Assets** tables will be populated or updated with the data originating from AutoCAD.

This data is populated in:

- The **General** tab, as well as in the **Portfolio** tab of a location detail.

  The AutoCAD drawing of a floor is visible in the **Document** tab of the floor of a location.
- The **General** tab, as well as in the **Documents** tab of an employee detail.
- The **General** and **Portfolio** tabs of a location detail.

Peregrine SYSTEMS

December 19, 2001