Automic™

Let's Automate Business.

Automation Engine - Naming Concepts

Guideline for the Naming of AE Objects

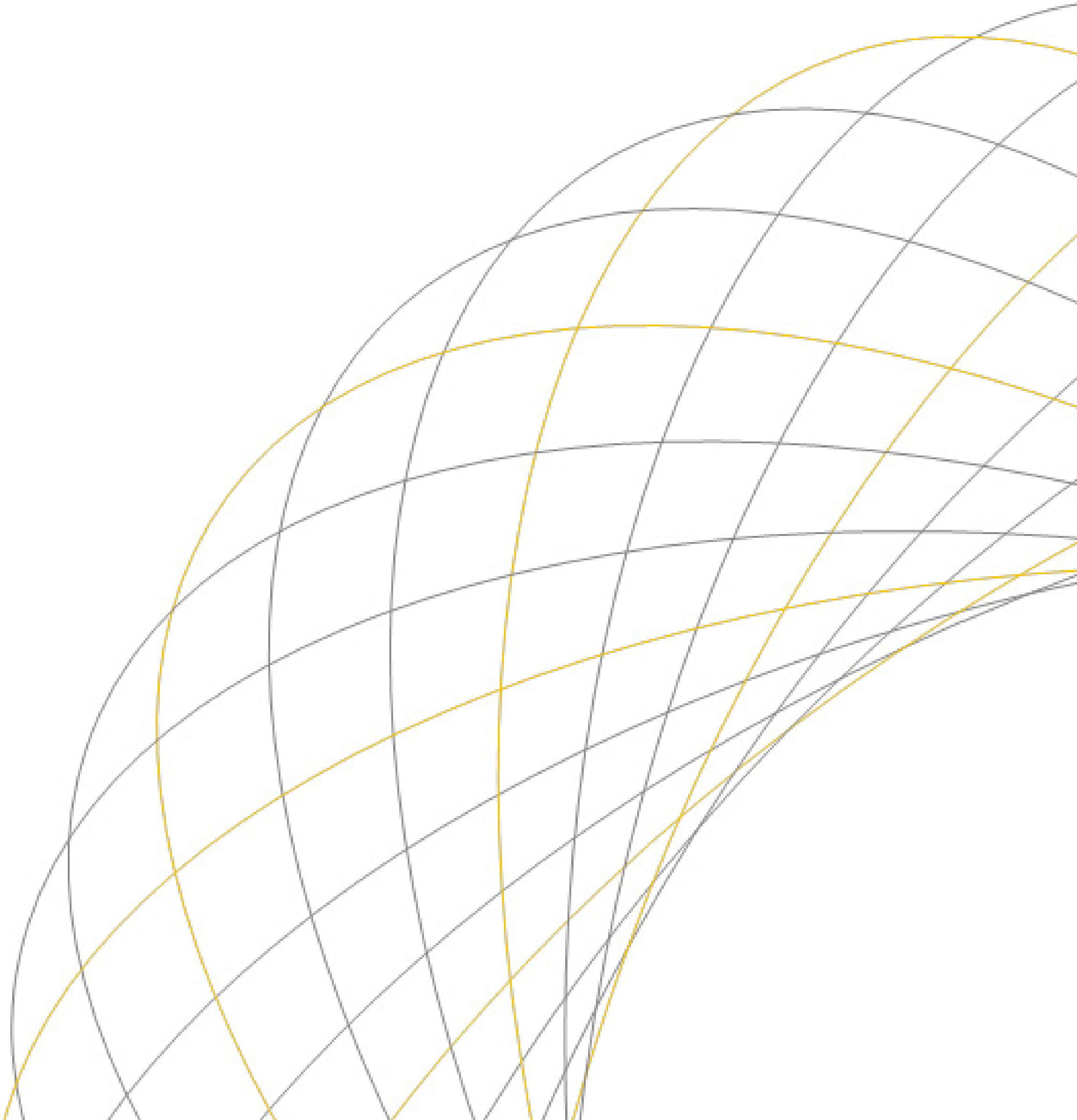


Table of Contents

Naming Concepts

1.1. Introduction

This document is intended to help you create naming concepts for your AE system. The definition of consistent naming concepts will greatly aid your daily use of the AE system, particularly in the following areas:

- Maintenance
- Alerts
- Security
- Transporting of objects and entire processes
- Training
- Revision

Naming concepts are relevant to the following objects and functions:

- Authorization concept
- Transport management
- Clients
- Agents
- Objects for the modeling of processes (e.g., Jobs, WorkFlows, Events, Variables, etc.)
- Folders and their structures
- Alerts

While it is possible to rename AE objects later on, this may prove to be very time-consuming. You should therefore carefully plan out your naming concept in advance.

1.2. Environment Concepts

Automic recommends to separate live environments from testing and development environments by using different servers and databases. This way, production will not be affected by test runs or updates to the testing environment. If needed, however, all three environments can be run within a single AE system.

1.3. Naming Concepts

This section introduces several different naming concepts and their practical use.

1.3.1. Character Conventions

AE object names may be up to 200 characters long. The following characters are permitted:

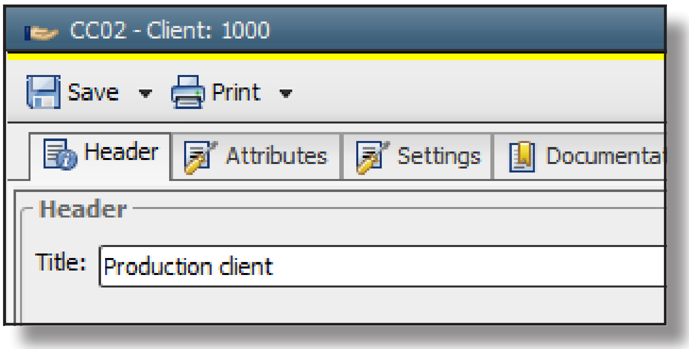
'A-Z' '0-9' '_' '.' '$' '@' '#'

The following objects have additional character length restrictions:

• Client	(4 numeric digits)
• Agent	(32 characters)
• AgentGroup	(32 characters)
• TimeZone	(8 characters)

1.3.2. Clients

An AE client acts as a self-contained environment. Clients are represented by client objects. The name of a client object needs to be a four-digit number between 0001 and 9999 (see fig. to the right).



Title: Production client

Figure 1.1

If there are no dependencies between the tasks (Jobs) for the individual subject areas/departments or customers, you may wish to run separate clients for them. The main advantage of this is an easier setup of the user authorization concept. On the other hand, merging clients later on can be a very time-consuming process.

The table below is an example of how to divide up the number ranges for your different areas. Note that the client numbers for each department or customer are uniformly different for the testing, development, and live systems. This type of numeric allocation is useful when it comes to defining tasks, as tasks are able to determine on which client, i.e., in which system they are running. Given a logical numbering concept, tasks can therefore configure the required parameters automatically. This can eliminate the need for manually configuring the tasks for each separate environment.

It also makes sense to create separate clients for the AE Database reorganization processes, as well as for other internal tasks such as training:

Test system	Development system	Live system
0001 training		
0100 internal maintenance	**0200** internal maintenance	**0300** internal maintenance
1000 FA Finances	**1100** FA Finances	**1200** FA Finances
2000 FA Insurances	**2100** FA Insurances	**2200** FA Insurances
3000 FA Warehousing	**3100** FA Warehousing	**3200** FA Warehousing

Table 1.1: Sample client concept

1.3.3. Agents

For the purposes of naming, there are two types of agents:

1.3.3.1. Agents for Operating Systems

A typical choice for naming individual operating system clients is to refer to the name of the system on which the agent is installed. Alternatively, the primary DNS name of the system can be used, should this not be the same as the system name. This way, you can immediately recognize within AE which system an operating system client is running on; this greatly aids the creation and allocation of operating system jobs. One potential disadvantage of this approach is that should the system be migrated to a new hardware system, the system name may also change. If the naming concept is to be preserved, this would require all the affected agent names to be updated, and with them the job object names. For jobs where the agent is entered directly in the object, this can be done via 'search & replace' in the AE user interface.

For dynamically generated jobs where the agent is only determined in runtime and then set by a script, renaming the agent object later on also can be time-consuming. In order for this not to become a major issue, the agent names should be stored centrally for each client or process (AE variables or Include objects). This approach especially makes sense if objects ever need to be transported from one client to another later on.

If the operating system (such as Unix, Windows, z/OS, etc.) is not stated as part of the name, we recommend to include it in the agent name as a prefix to the system name. If you choose this approach, make sure all your prefixes have a uniform length (e.g., UNX_, WIN_, ZOS_, etc.).

1.3.3.2. Agents for Applications

To better distinguish agents for applications from agents for operating systems, these should be named with a set prefix that uniquely identifies the application, followed by the name of the application instance:

Structure: <APP-PREFIX>_<APP_INSTANCE>
Example: SAP_P01

1.3.4. Agent Groups

Agent groups encapsulate agents that use the same operating system or application. In Jobs, agents groups can be specified instead of individual agents for the purposes of load balancing or process parallelization, or to ensure high availability. Naming should be based on the shared traits of the agents encapsulated by the group. To help distinguish them and to aid searches of agent groups, you may wish to use a prefix such as 'AG_':

Object name	Description
AG_UNX_ALL	Group consisting of all UNIX agents
AG_UNX_ORA	Group consisting of all UNIX agents running on Oracle database servers
AG_WIN_SQL	Group consisting of all Windows agents running on MSSQL database servers
AG_SAP_BASIS	Group consisting of all SAP agents to be used for running SAP Basis jobs
...	

Table 1.2: Example of how to name agent groups

1.3.4.1. Clustered Agents

If agents are being run as part of a cluster, we recommend to use a prefix or suffix to distinguish these from non-cluster agents.

Object name	Description
P01_UNX_CLUSTER_A	UNIX agent for SAP P01 instance in cluster ‚A‘
P02_WIN_CLUSTER_X	Windows agent for SAP P02 instance in cluster ‚X‘
...	

Table 1.3: Naming agents that are part of a cluster

1.3.5. Objects for Process Modeling

Objects used for process modeling are either process-related or global objects. Examples of global objects are Login, Include, Variable, Notification, and Calendar objects. These can be used by all of the processes within a client. Examples of process-related objects are Jobs, FileTransfers, and WorkFlows.

1.3.5.1. General Considerations

When choosing object names, it is very important to pay attention to the access authorizations. An authorizations concept that is both effective and easy to maintain relies on a logically structured naming concept. When assigning names, also make sure to take into consideration the transport paths that will be used. For example, should two clients be merged into a single client in the future, objects with identical names will cause issues.

If there is any intention to transport processes between AE clients, such as between a testing client and a live client, all of the client-dependent parameters for each self-enclosed process (transport unit) should be stored at a central process location (e.g., AE Variable object). This includes agents and paths for processing that may vary between testing and production.

This central configuration object either contains the configuration for all the clients running the process, in which case it is always included in the transport, or it only contains the configuration of the client currently running the client, in which case it may not be part of the transport. The first of these two approaches is more time-consuming in terms of maintenance, but it is also more reliable for operation.

In AE, authorizations can only be controlled via the object names and object types; they cannot be controlled via user-defined folder structures. Because of this, global objects that are only accessible for modification by a restricted user group should be identified accordingly in naming.

An object name should therefore consist of the following:

- **Static part of name** (depending on object type) at the start of the object name, for authorization verification. Examples of what may be included here:

  - Branch
  - Department
  - Special identifier for a client's global objects
  - AE object type

  The individual identifiers of the static part of the name should have a uniform structure and length for each object type. This aids readability and makes it easier to find objects, and also simplifies automatic evaluation of object names in the case of dynamic objects. Make sure the static part of the name is not too long; this also aids readability, for example, in process schedules that only display the start of the object names.

- **Dynamic part of name** at the end of the name to individually describe the task encapsulated by the object.
-
- Unique, uniform**separator** between the static and dynamic parts of the name.

Sample structure of object name	Object type (UC4 short form)
SYS.<**CCCC**>#<**description**>	Globally valid objects within the client
<**AA**>.<**BBBB**>.<**CCCC**>_<**DDD**>#<**description**>	Job (JOBS)
<**AA**>.<**BBBB**>.<**CCCC**>#<**description**>	WorkFlow (JOBP)
<**AA**>.<**BBBB**>.<**CCCC**>#<**description**>	Script (SCRI)
<**AA**>.<**BBBB**>.<source>.<target>.<**CCCC**>#<**description**>	FileTransfer (JOBF)
...	...

Table 1.4: Sample naming concept for individual object types

Key:

SYS	identifies an object that can be used globally across the client; modification is restricted to a specified user group
<**AA**>	country code (two-digit, e.g., ISO 3166-compliant)
<**BBBB**>	department (four-digit)
<**CCCC**>	AE short form for object type

<DDD>	platform (three-digit)
<description>	User text for describing the process task
#	Separator character between static and dynamic parts of the name

As part of the naming concept, each object type should be given a uniform structure.

Object name (examples)	Description
SYS.LOGIN#WINDOWS	Global Login object for all Windows agents
SYS.CALE#PRODUCTION	Global calendar for production
UK.FIAC.JOBS_UNX#MONTH_END_CLOSE$STEP1	Unix job step 1 for monthly settlement of Financial Accounting Dept. in UK
UK.FIAC.JOBP#MONTH_END_CLOSE$MASTER	Master WorkFlow for monthly settlement of Financial Accounting Dept. in UK
UK.FIAC.SCRI#MONTH_END_CLOSE$CHECK_FILES	File checking script for monthly settlement of Financial Accounting Dept. in UK
...	

Table 1.5: Object naming examples

1.3.6. Using Global Objects

To simplify the maintenance of processes, the use of global objects should be instated right at the beginning, i.e., before any of the processes are transferred to the live system. Central objects, for example, can already be planned for as part of the object templates when creating new processes.

Examples for this are global Includes (script components) for all the objects' available script cards, which later on permit fast updating of multiple objects, or the definition of at least one central Notification object for terminations within a process. Both examples would require not only the corresponding objects to be defined but also for the object template(s) to be updated in order for the central objects to available for all new processes. Any existing process tasks still based on the old templates would need to be adapted manually. Resultantly, it makes sense to plan out the use of global objects as early as possible.

Object name (examples)	Description
SYS.INC#JOBS_UNX.PRE	Include for the pre-script card of the UNIX job template
SYS.INC#JOBS_UNX.SCRI	Include for the script card of the UNIX job template
SYS.INC#JOBS_UNX.POST	Include for the post-script card of the UNIX job template
SYS.CALL#OPERATING	Include for the attribute card of the WorkFlow template (‚Result evaluation per single task‛)
...	

Table 1.6: Example of how to name and use global objects

1.3.6.1. Using Object Names for Dynamic Changes

'Executable objects' such as Jobs or FileTransfers are able to determine their own name when launched. Any useful information contained in the name can then be used to configure the object's parameters.

Some Automic customers are making great use of this approach, particularly in the area of SAP processing. For example, the object name is augmented by the name of the SAP report and its associated variant:

Object name (syntax)	Description
<AA>.<BBBB>.<CCCC_DDD>#<description>	General Jobs
<AA>.<BBBB>.CCCC_DDD>#<report>@<variant>#<description>	SAP Jobs
...	

Table 1.7: Examples of dynamic object naming

The AE script commands for dissecting the name are again stored in central Include objects. These Includes are added to the (SAP) Job templates right at start of constructing the AE system. This way, every newly created Job already has the Include required for it.