SAP NetWeaver Security and Identity Management

SAP Identity Management APIs

Document Version 1.20 – December 2009

SAP AG
Dietmar-Hopp-Allee 16
69190 Walldorf
Germany
T +49/18 05/34 34 24
F +49/18 05/34 34 20
www.sap.com

Typographic Conventions

Type Style	Represents
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Contents

Identity Management APIs

Purpose

As part of the SAP identity management model, we provide a set of application programming interfaces (APIs) that can be used by identity and access management (IAM) vendors to manage SAP users and role assignments within their systems.

The APIs exist for both ABAP and Java systems. The ABAP APIs are provided as Business Application Programming Interfaces (BAPIs). The Java APIs are provided with the user management engine (UME) interfaces.

This documentation provides an overview of how to use the BAPIs and the Java APIs for identity management. It does not cover all of the details, but shows the general approach and highlights those aspects where special considerations are necessary. For details about using specific APIs, see the API documentation.

Prerequisites

Before using this document and the corresponding APIs, you should be familiar with the SAP identity management concept. For more information, see Identity Management of the Application Server ABAP [SAP Library] and Identity Management of the Application Server Java [SAP Library].

Integration

We also offer a central user administration (CUA) for managing SAP users and role assignments for ABAP-based SAP systems. When determining how to integrate an external IAM system into a landscape that uses the CUA, we recommend the following scenarios:

- The customer can connect the IAM system to the CUA central system, which in turn provides the user and role assignment information to its child systems.
- The customer can connect the IAM system to each of the child systems. In this case, the customer should remove the SAP CUA from the system landscape.

  Customers should not connect the external IAM system to the CUA child systems and continue using the CUA. This can lead to discrepancies.

About this Document

This documentation is divided into the following sections:

- Identity Management ABAP APIs
  - Concepts
    - BAPI Explorer
    - Business object: USER
    - Authorization Assignments
  - APIs to use for User Administration Functions in ABAP
  - Integration with Central User Administration
  - Appendix
    - Overview of the IAM BAPIs and function modules
    - Sample parameters for BAP_HELPVALUES_GET
    - Includes used by the function module IAM_API_TESTFRAME
- Identity Management Java APIs
  - APIs to use for User Administration Functions in Java
  - Appendix: Schema Description

1 Identity Management ABAP APIs

1.1 Purpose

This section describes how to use the ABAP identity management APIs for managing users and roles. This API consists primarily of remote functions calls that are available as BAPIs in the ABAP system.

This documentation does not describe the details of each API as these are documented in the system; instead it introduces the concepts and highlights those aspects that need special consideration.

The main functions available with the ABAP API are demonstrated in the example program IAM_API_TESTFRAME. Each function is available as a program include file which is accessible from the test program using a simple user interface.

For more information about the individual BAPIs, see the corresponding system documentation, which is available using the BAPI Explorer [Page 4] (transaction BAPI).

1.2 Integration

Systems that use the SAP ABAP-based user management can use the Central User Administration (CUA) to maintain and distribute user and role information to the individual systems. There are certain aspects that need to be taken into account when the APIs are used in CUA landscapes. These are described in Integration With Central User Administration [Page 19].

1.3 Concepts

Before beginning with the APIs, you should be familiar with some of the concepts that apply to SAP's identity management technology and how to use the BAPIs. See:

- BAPI Explorer [Page 4]

  This topic introduces the BAPI Explorer, which you can use to view and maintain BAPIs in the ABAP system.

- Business Object: USER [Page 4]

  This topic describes the structure of the *USER* business object and how it relates to the *AddressOrg* business object.

- Authorization Assignments [Page 6]

  This topic introduces the various methods available for assigning authorizations, for example, using a reference user and assigning roles or profiles to users.

- Role Assignment for IAM Administrator [Page 7]

  This topic provides the role to use to assign the authorizations for IAM administration using the BAPIs.

1.3.1 BAPI Explorer

1.3.1.1 Use

BAPIs are remote function calls (RFCs) that represent an object-oriented view of business objects. The BAPI module accesses the corresponding method that applies to the object.

For example, the RFC module BAPI_USER_GET_DETAIL implements the *GetDetail()* method for the business object *USER*.

To see the BAPI representation for the business objects, use the BAPI Explorer (transaction BAPI). With the BAPI Explorer, you can view and maintain business objects and their methods. You can also find the detailed documentation for the objects, methods, and corresponding parameters.

For more information about using the BAPI Explorer, see the BAPI Explorer [SAP Library] documentation. If the system is set up to use the online help, you can access this documentation by choosing *Help* → *Application Help* from the BAPI Explorer menu.

1.3.2 Business Object: USER

1.3.2.1 Definition

Business object that corresponds to the user who logs on to the system.

1.3.2.2 Use

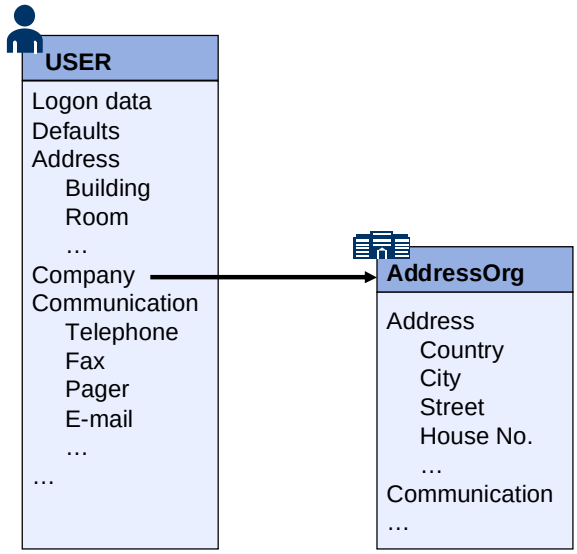
The most important business obect used by the identity management APIs is the business object *USER*.

USER contains the technical information used for logon and work in the SAP system, for example, validity, role and profile assignments and printer settings. In addition, it also contains communication data such as telephone number, fax, e-mail address, company assignment and additional address data.

1.3.2.3 Structure

The following figure shows the structure of the *USER* business object and how it relates to the *AddressOrg* business object, which contains the company address.

***USER* and *AddressOrg* Business Objects Attributes**



The business object *USER* contains the logon data, default settings, address and communication data. It also contains a read-only reference to the *AddressOrg* business object.

Because this is a read-only reference, you cannot change the user's company address data using the *USER* BAPIs. You can only change the user's assignment to the company.

1.3.2.4 Methods

The methods used by the IAM API are described in more detail in the rest of this documentation. For a list of all of the methods available, see the *USER* object in the BAPI Explorer.

1.3.2.5 Integration

The business object *USER* only contains the technical information about the SAP system user, but not the user as a business partner. See the SAP Business Partner [SAP Library] documentation for more information about obtaining this information.

1.3.3 Authorization Assignments

1.3.3.1 Use

To assign authorizations to users, you can use any of the following approaches:

- Reference users
- Roles
- Profiles

You can use these approaches in parallel, for example, you can assign a reference user as well as roles and profile to a user.

1.3.3.2 Reference User

When using the reference user, the administrator assigns authorizations to a reference user rather than directly to the user. Users then point to the reference user. During an authorization check, the system first checks the reference user's authorizations, and then those of the user.

This procedure is best suited to situations involving large numbers of users of the same type.

For example, an online shop hosts a large number of users that are to have the same authorizations, for example, to browse in the product catalog, order the contents of a shopping basket, or check the status of an order. Therefore, all of the authorizations required for these functions are assigned to a reference user, which is then referenced by the individual users.

Without this procedure, the administrator would have to assign the same authorizations to every user. For systems with a large number of users, this can lead to significant performance problems, as the user administration tools then have to handle large amounts of data that are redundant.

A reference user is assigned when the user is created, but you can also change this assignment later by modifying the user.

1.3.3.3 Role Assignment

With this procedure, individual rights are assigned to the user using one or several roles. Unlike the reference user, this type of authorization assignment is very flexible. You can group together authorizations for common tasks in a few large roles and create further smaller roles with individual extensions.

In addition to authorizations, roles can hold additional user settings, for example user menus provided by the SAP GUI for Windows Easy Access Menu.

If you are dealing with a very large number of users, we recommend using reference users, even if additional roles are also required.

Roles are assigned after user creation.

1.3.3.4 Profile Assignment

Profiles are similar to roles in that they are containers for authorizations. They are assigned in the same way. However, profiles are older and unlike roles, they cannot contain additional information, therefore, using roles is the preferred method for assigning authorizations.

1.3.4 Role Assignment for IAM Administrator

The administrator or service user who calls the IAM API function needs the authorizations for performing user and role maintenance functions remotely. These authorizations are included in the role SAP_BC_USR_CUA_CLIENT. Therefore, use this role assignment for the user that calls the IAM API functions.

1.4 APIs for User Administration Functions in ABAP

This section provides an overview of the user administration functions required by an Identity and Access Management (IAM) system. The modules that are available for implementation for the following functions are described. See the following topics:

- BAPIs for User Enquiry [Page 8]
- BAPI for Creating Users [Page 9]
- BAPI for Modifying Users [Page 11]
- BAPI for Deleting Users [Page 12]
- BAPIs for Setting Passwords [Page 13]
- BAPIs for Locking and Unlocking Users [Page 14]
- BAPIs for Obtaining or Maintaining Company Information [Page 15]
- BAPIs for Role Assignment [Page 16]
- BAPIs for Profile Assignment [Page 17]

There are contraints that you consider when the IAM system is to be used in a landscape where Central User Administration (CUA) is also used. These considerations are described in the section Integration With Central User Administration [Page 19].

1.4.1 BAPIs for User Enquiry

The following functions and their corresponding BAPIs provide information about a user.

1.4.1.1 Obtaining a List of Users

1.4.1.1.1 BAPI: BAPI_USER_GETLIST

Business object method: *USER.GetList()*

Use: As of Release 6.20, support package 38, you can use this BAPI to retrieve a list of users that match complex selection criteria. The use of wildcards in the search is supported. For the output format, you can you can select either user ID only or user ID with first, last and complete names.

Examples of selection criteria include assigned roles or lock status. As of SAP NetWeaver 2004s, you can also search using the last modification date.

In older releases, use the value help function BAPI_HELPVALUES_GET to obtain a list of users.

For an example, see the test program IAM_API_TESTFRAME. Enter the search criteria and select the option *Userlist* or *Userlist with names*.

The corresponding include provided with the test program is IAM_USERLIST.

1.4.1.2 Obtaining Detailed Information About a User

1.4.1.2.1 BAPI: BAPI_USER_GET_DETAIL

Business object method: *USER.GetDetail()*

Use: Use this BAPI to obtain information about a specific user, for example, logon data, default parameters, communication information, the user's company address, and the user's assigned roles. As of SAP NetWeaver Release 2004s, a user's lock status is also returned.

For older releases, use SUSR_USER_LOCKSTATUS_GET and SUSR_LOGIN_CHECK_RFC to obtain the lock status.

You must provide the user ID for the user you want to search for. Wildcards are not supported for the search.

For an example, see the test program IAM_API_TESTFRAME. Select the option *User details* for a user to obtain the detailed information.

The corresponding include provided with the test program is IAM_USERDETAILS.

1.4.1.3 Getting Value Help for User Parameters

1.4.1.3.1 BAPI: BAPI_HELPVALUES_GET

Business object method: *Helpvalues.GetList()*

Use: This BAPI is an all-purpose BAPI that is also used by the IAM API to retrieve value help for user parameters. A list of example parameters for using this BAPI to obtain user information is included in the appendix under Parameters for BAPI_HELPVALUES_GET [Page 25].

For many cases, it is more appropriate to use the designated BAPI to obtain information such as:

- List of users:BAPI_USER_GETLIST (as of Release 6.20, support package 38)
- List of roles: PRGN_ROLE_GETLIST (as of SAP NetWeaver 2004s)
- Company address: BAPI_ADDRESSORG_GETDETAIL

For an example, see the test program IAM_API_TESTFRAME. Select the option *Company list*. This option uses the BAPI_HELPVALUES_GET module to obtain a list of companies. Although this option shows the use of the BAPI for a specific parameter, you can use it as an example for obtaining value help for other user parameters.

The corresponding include provided with the test program is IAM_COMPANYLIST.

1.4.2 BAPI for Creating Users

1.4.2.1.1 BAPI: BAPI_USER_CREATE1

Business object method: *USER.Create1()*

Use: Use this BAPI to create users in the ABAP system. These users are maintainable using the transaction SU01. To craete a user, call BAPI_USER_CREATE1 with the parameters listed below.

- *User name*: The user name is a required parameter.
- *Validity Period of the User*: Enter the validity period in the *LOGONDATA* structure. This structure contains the fields for valid from (*GLTGV*) and valid to (*GLTGB*).
- *Initial password*: The initial password is a required parameter. The user must change this password the first time he or she logs on. You can either specify the initial password directly or generate a random one. To generate a random password, use the function module RSEC_GENERATE_PASSWORD.

This function module cannot be called remotely.

- *Address Data*: The field *LASTNAME* is a required field; all other fields are optional. The table parameters of the function module are also optional.

  Although there are additional address data fields, only those fields that are relevant for maintenance using transaction SU01 are stored in the user master record.

- *Reference User*: This parameter is optional. If applicable, provide the user ID to use as a reference user.

Next Steps: Once you have created the user, you may need to perform any the following actions:

- Assign roles: For information about how to assign roles to the user, see BAPIs for Role Assignment [Page 16].

- Lock the user: You can lock the user if it should be available at a later date, for example, after being checked by the departmental manager. For more information, see BAPIs for Locking and Unlocking Users [Page 14].

- Set a productive password: The user is created with an initial password that needs to be changed the first time the user logs on. However, there may be cases where you want to set up a user that initially has a productive password. Technically this is not possible, therefore in this case, you have to create the user with an initial password and then change it using the function module SUSR_USER_CHANGE_PASSWORD_RFC. For more information, see BAPIs for Setting Passwords [Page 13].

  For an example on creating users, see the test program IAM_API_TESTFRAME. Enter the data for the user and select the option *Create user*.

  The corresponding include provided with the test program is IAM_USERCREATE.

  In the test program IAM_API_TESTFRAME, the user's initial password is set to **`Initial`**.

1.4.3 BAPI for Modifying Users

1.4.3.1.1 BAPI: BAPI_USER_CHANGE

Business object method: *USER.Change()*

Use: Use this BAPI to modify users in the ABAP system. *USER.Change()* has a similar structure and table parameters as *USER.Create1()*. In addition, each modifiable parameter has a corresponding flag parameter that specifies which data is to be changed. Structures have flags for each field. Table parameters have a flag for each column.

For example, for the parameter *Logondata*, there is a corresponding flag parameter *Logondatax*.

When changing data, consider the following special cases:

- **Address:** You can maintain certain address data in the *Address* structure or alternatively in tables. For example, data such as telephone number, fax and e-mail address can be maintained in the tables *AddTel*, *AddFax*, and *AddSmtp* respectively.

  We recommend maintaining the information in the tables instead of in the *Address* structure for the following reasons:

  - You can store multiple entries in the tables. The *Address* structure only contains one entry for each of these fields.
  - The telephone and fax numbers are stored in international format in the tables, but not in the *Address* structure.
  - If you change data in the *Address* structure, any entries in the corresponding table will be lost.

- **Communication data:** When changing communication data (*Add<Xxx>* parameters), you need to consider the following fields:

  - *CONSNUMBER:* To differentiate between multiple entries for communication data, use the sequence number that is stored in the field *CONSNUMBER*. To change a specific entry, enter the entry's sequence number in this field. If you want to add an entry, specify a sequence number that is higher than that for any existing entry.
  - *R_3_USER*: This field applies to the telephone numbers. It indicates the type of telephone connection and if the number used is the standard number. The following applies:
    - **<blank>** : The telephone number is a land-line telephone.
    - **1** : The telephone number is the standard land-line telephone.
    - **2** : The telephone number is a mobile telephone.
    - **3** : The telephone number is the standard mobile telephone.
  - *STD_NO*: Only one telephone number appears as the standard telephone number in the *Address* structure. Therefore, use this field to indicate that the telephone number (land-line or mobile) for this entry is the overall standard telephone number that appears the *Address* structure.

- *STD_RECIP*: This field indicates whether the corresponding telephone number can be used for short messages (SMS). If this is the case, then the number is copied to the communication data used for paging services.

Not all fields are used by all of the communication data parameters.

- **Company Location:** The company location address is stored with the business object *AddressOrg* and not the object *USER*. Therefore, when specifying or changing the company location with the BAPI_USER_CHANGE, you can only specify or assign an existing company location. To change the company address, use the methods provided for the business object *AddressOrg*. For more information, see BAPIs for Obtaining or Maintaining Company Information [Page 15].

For an example on using BAPI_USER_CHANGE module, see the test program IAM_API_TESTFRAME. Select the option *Modify user* for a specific user to change his or her parameters.

The corresponding include provided with the test program is IAM_ USERCHANGE.

- **Table parameters GROUP and PARAMETER1:** The flags for these table parameters are set up according to the columns in each table. Set the flag parameter for the first column to indicate changes in the table parameter.

PARAMETER1 replaces the table parameter PARAMETER.

For PARAMETER1, the contents of the field PARTXT cannot be changed using BAPI_USER_CHANGE.

1.4.4 BAPI for Deleting Users

1.4.4.1.1 BAPI: BAPI_USER_DELETE

Business object method: *USER.Delete()*

Use: Use this BAPI to delete users.

Because additional data is associated with users, for example, change documents, you should consider deactivating users instead of deleting them. To deactivate a user, either lock it or specify a validity period.

No license fees are collected for deactivated users.

For an example on deleting users, see the test program IAM_API_TESTFRAME. Select the option *Delete user* for a specific user.

The corresponding include provided with the test program is IAM_USERDELETE.

1.4.5 BAPIs for Setting Passwords

1.4.5.1 Setting an Initial Password

1.4.5.1.1 BAPI: BAPI_USER_CHANGE

Business object method: *USER.Change()*

Use: Use BAPI_USER_CHANGE to set an initial password. Set the initial password in the *Password* field and set the flag *Passwordx*. The user must change this password the next time he or she logs on.

A password set when creating a user with BAPI_USER_CREATE1 is also set as an initial password and must be changed when the user logs on for the first time.

The test program IAM_API_TESTFRAME does not support setting the initial password.

1.4.5.2 Setting a Productive Password

1.4.5.2.1 BAPI: BAPI_USER_CHANGE

Business object method: *USER.Change()*

Use: Use BAPI_USER_CHANGE to set a productive password using the optional parameter *Productive_Pwd*.

This parameter was introduced with the following support packages.

- Release 7.0 SPS 19
- Release 7.0 EhP1 SPS 5
- Release 7.1 SPS 10
- Release 7.1 EhP1 SPS 5

For more information and the corresponding correction instructions, see SAP Note 1287410.

1.4.5.3 Changing Passwords

1.4.5.3.1 Function Module: SUSR_USER_CHANGE_PASSWORD_RFC

Use: You cannot use BAPI_USER_CHANGE to change a password. For this purpose, you can use the function module SUSR_USER_CHANGE_PASSWORD_RFC. This module requires the old and the new passwords.

If you use the RFC SDK, then we recommend using the function modules RfcOpenEx() and RfcRegisterPasswordChanger() that are available with the RFC API to change the password and to have the SAP system report an expired password, respectively.

1.4.6 BAPIs for Locking and Unlocking Users

1.4.6.1 Locking users

1.4.6.1.1 BAPI: BAPI_USER_LOCK

Business object method: *USER.Lock()*

Use: Use this BAPI to lock users.

For an example, see the test program IAM_API_TESTFRAME. Select the option *Lock user* for a specific user.

The corresponding include provided with the test program is IAM_USERLOCK.

1.4.6.2 Unlocking users

1.4.6.2.1 BAPI: BAPI_USER_UNLOCK

Business object method: *USER.Unlock()*

Use: Use this BAPI to unlock users.

For an example, see the test program IAM_API_TESTFRAME. Select the option *Unlock user* for a specific user.

The corresponding include provided with the test program is IAM_USERLOCK.

The test program returns a message that the user is unlocked if this is permitted. This applies to systems where CUA is used. If CUA is used, there may be cases where unlocking a user in a child system is not possible because of a global lock. For more information, see Considerations for User Administration Functions [Page 21].

1.4.7 BAPIs for Obtaining or Maintaining Company Information

The following functions and their corresponding BAPIs provide information about a company.

1.4.7.1 Obtaining a List of Companies

1.4.7.1.1 BAPI: BAPI_HELPVALUES_GET

Business object method: *Helpvalues.GetList()*

Use: This BAPI is an all-purpose BAPI that is also used by the IAM API to retrieve value help for user and company parameters. A list of the parameters for this BAPI are listed in the appendix under Parameters for BAPI_HELPVALUES_GET [Page 25].

For an example, see the test program IAM_API_TESTFRAME. Select the option *Company list* to obtain a list of available companies. The use of wildcards in the search is supported.

The corresponding include provided with the test program is IAM_COMPANYLIST.

This example uses BAPI_HELPVALUES_GET, to obtain a list of available companies. The values for the list are returned in the parameter *HELPVALUES*. The parameter *DESCRIPTION* returns metadata, which, in this example, is used for the list headings.

You can use BAPI_HELPVALUES_GET to obtain help for other user parameters as well.

1.4.7.2 Obtaining Detailed Information About a Company

1.4.7.2.1 BAPI: BAPI_ADDRESSORG_DETAIL

Business object method: *AddressOrg.FindDetail()*

Use: Use this BAPI to obtain company address information. It can serve as a value help to obtain the company location information for a user.

For an example, see the test program IAM_API_TESTFRAME. Select the option *Company detail* for a specific company (field *Company*) to obtain the detailed information. The use of wildcards in the search is not supported.

The corresponding include provided with the test program is IAM_COMPANYDETAILS.

1.4.7.3 Modifying Address Data for a Company

1.4.7.3.1 BAPI: BAPI_ADDRESSORG_CHANGE

Business object method: *AddressOrg.Change()*

Use: Use this BAPI to change company address information.

The test program IAM_API_TESTFRAME does not provide an example for using this BAPI.

1.4.8 BAPIs for Role Assignment

1.4.8.1 Obtaining a List of Roles

1.4.8.1.1 Function Module: PRGN_ROLE_GETLIST (or BAPI_HELPVALUES_GET)

Use: As of SAP NetWeaver Release 2004s, you can use the function module PRGN_ROLE_GETLIST as a value help to obtain a list of roles. It is implemented as an RFC-enabled function module, not as a BAPI. Alternatively, you can use BAPI_HELPVALUES_GET to obtain this information, however, the function module PRGN_ROLE_GETLIST also supports the use of wildcards and ranges.

Prior to Release 2004s, use BAPI_HELPVALUES_GET to obtain this information.

For an example, see the test program IAM_API_TESTFRAME. Select the option *Role list* for a role search pattern (enter the pattern in the *Role* field). The use of wildcards in the search is supported.

The corresponding include provided with the test program is IAM_ROLELIST.

1.4.8.2 Obtaining a List of Role Assignments

1.4.8.2.1 BAPI: BAPI_USER_GET_DETAIL

Business object method: *USER.GetDetail()*

Use: Use this BAPI to obtain information about a user, which includes the user's role assignments.

For an example, see the test program IAM_API_TESTFRAME. Select the option *Roles of a user* for a specific user. The use of wildcards in the search is not supported.

The corresponding include provided with the test program is IAM_USERROLES.

1.4.8.3 Assigning Roles

1.4.8.3.1 BAPI: BAPI_USER_ACTGROUPS_ASSIGN

Business object method: *USER.ActgroupsAssign()*

Use: Use this BAPI to assign roles. Note however, that if you want to change a user's role assignments, you must first use BAPI_USER_GET_DETAIL to obtain the user's role assignments. You can then add or remove roles and then set the new role assignment using BAPI_USER_ACTGROUPS_ASSIGN. The system then replaces the old role assignments with the new ones.

Fields *FROM_DAT* and *TO_DAT*: If these fields are not set, then *FROM_DAT* is set to the current date and *TO_DAT* to December 31, 9999.

For an example, see the test program IAM_API_TESTFRAME. Select the option *Assign roles* for a specific user and role you want to assign.

The corresponding include provided with the test program is IAM_ROLEASSIGN.

1.4.8.4 Deleting Role Assignments

1.4.8.4.1 BAPI: BAPI_USER_ACTGROUPS_DELETE

Business object method: *USER.ActgroupsDelete()*

Use: Use this BAPI to delete all role assignments. The same result occurs if you use BAPI_USER_ACTGROUPS_ASSIGN and pass an empty table of roles.

This function deletes all role assignments. If you only want to delete some of the role assignments, modify the role assignments using BAPI_USER_ACTGROUPS_ASSIGN.

For an example, see the test program IAM_API_TESTFRAME. Select the option *Delete role assignments* for a specific user.

The corresponding include provided with the test program is IAM_ROLEASSIGN_DELETE.

1.4.9 BAPIs for Profile Assignment

Although we recommend using roles to assign authorizations to users, it is possible and sometimes necessary to still use profiles. For example, you may need to maintain authorizations based on already existing profiles that have not been migrated to roles, or you may want to assign authorizations using the SAP-delivered profiles such as SAP_NEW (or SAP_ALL). The following BAPIs are provided for maintaining authorizations using profiles.

1.4.9.1 Obtaining a List of Profiles

1.4.9.1.1 BAPI: BAPI_HELPVALUES_GET

Use: To obtain a list of profiles, use the general value help BAPI_HELPVALUES_GET. You can obtain a list of active single or composite roles, or profiles that have been generated from roles. The corresponding parameter settings are shown in the table below.

Parameters for BAPI_HELPVALUES_GET to Obtain a List of Profiles

Parameter	Value
OBJTYPE	`USER`
OBJNAME	`<blank>`
METHOD	`GETDETAIL`
PARAMETER	`PROFILES`
FIELD	`<blank>`

EXPLICIT_SHLP-SHLPNAME	`PROFILES_SINGLE_ACTIVE`
	`PROFILES_COMPOSITE_ACTIVE`
	`PROFILES_GENERATED_ACTIVE`

EXPLICIT_SHLP-SH	`SH`

For an example, see the test program IAM_API_TESTFRAME. Select the option *Profile list*. The use of wildcards in the search is supported.

The corresponding include provided with the test program is IAM_PROFILELIST.

1.4.9.2 Obtaining a List of Profile Assignments

1.4.9.2.1 BAPI: BAPI_USER_GET_DETAIL

Business object method: *USER.GetDetail()*

Use: Use this BAPI to obtain information about a user, including the profile assignments.

For an example, see the test program IAM_API_TESTFRAME. Select the option *User details* for a specific user. The use of wildcards in the search is not supported. The user's assigned profiles are included in the user's data obtained by this call.

The corresponding include provided with the test program is IAM_USERDETAILS.

1.4.9.3 Assigning Profiles

1.4.9.3.1 BAPI: BAPI_USER_PROFILES_ASSIGN

Business object method: *USER.ProfilesAssign()*

Use: Use this BAPI to assign profiles to users. All profiles specified in the corresponding parameters are assigned to the user. Therefore, to change a user's profile assignments, you must first use BAPI_USER_GET_DETAIL to obtain the user's profile assignments. You can then add or remove profiles and then set the new profile assignment using BAPI_USER_PROFILES_ASSIGN. The system then replaces the old profile assignments with the new ones.

Example: There is no example for profile assignments provided with the test program IAM_API_TESTFRAME. However, the call is similar to the call for assigning roles.

1.4.9.4 Deleting Profile Assignments

1.4.9.4.1 BAPI: BAPI_USER_PROFILES_DELETE

Business object method: *USER.ProfilesDelete()*

Use: Use this BAPI to delete all profile assignments. The same result occurs if you use BAPI_USER_PROFILES_ASSIGN and pass an empty table of profiles.

There is no example for profile assignments provided with the test program IAM_API_TESTFRAME. However, the call is similar to the call for deleting role assignments.

1.5 Integration With Central User Administration

As previously mentioned, there are issues to consider when integrating an external IAM system in a landscape where the central user administration (CUA) is used for distributing SAP user information and role assignments.

When using CUA, the customer sets up a system where the administrator maintains the SAP user attributes and role assignments centrally and then has this information distributed to the CUA's child systems.

1.5.1 Recommendations

When determining how to integrate an external IAM system into a landscape that uses the CUA, we recommend the following scenarios:

- The customer can connect the IAM system to the CUA central system, which in turn provides the user and role assignment information to its child systems.
- The customer can connect the IAM system to each of the child systems. In this case, the customer should remove the CUA from the system landscape.

Customers should not connect the external IAM system to the CUA child systems and continue using the CUA. This can lead to discrepancies.

1.5.2 Maintaining Attributes Globally or Locally

Depending on the configuration, the customer can maintain most of the user attributes either globally in the central system or locally in the child systems. For the possible settings and the corresponding effects, see the table below.

CUA Settings for Distributing Attributes

Setting	Effect
Global	The attribute is maintained in the central system and distributed to the child systems.
Local	The attribute is maintained in the child system and is not distributed.
Default	A default value is maintained in the central system and distributed to the child system when a user is created. Afterwards, the value is maintained in the child system.
Redistribution	The attribute can be maintained either in the central system or in the child system. If the attribute is changed in a child system, it is also changed in the central system and redistributed to all child systems.
Everywhere	The attribute can be maintained either in the central system or in the child systems. No redistribution occurs. This setting is not available for all attributes.

The only settings that are relevant in regard to the IAM APIs are *global* and *local*. The other settings are derivations from these.

1.5.3 Considerations When Using the IAM APIs for a CUA System Landscape

Therefore, when using the IAM APIs, you need to consider these possible distribution strategies when creating or changing user attributes. Attributes that are specified to be maintained globally can be maintained by the external IAM system and then distributed by the CUA to the child systems. However, when attributes that are specified to be maintained locally are changed in the external IAM system, then changes are only propagated to the CUA central system. They are not distributed to child systems.

These considerations apply for cases where the customer connects the external IAM system to the CUA central system. If the customer connects the IAM system to the child system(s), then he or she should no longer use the CUA for the distribution of user information.

The considerations to take into account for the individual APIs are described in the sections that follow. See:

- Considerations for User Administration Functions [Page 21]
- Considerations for Role and Profile Assignment Functions [Page 22]

1.5.4 Considerations for User Administration Functions

1.5.4.1 Creating a User

The procedure for creating a user in a CUA landscape is the same as in an individual system (see BAPI for Creating Users [Page 9]). There are no extensions to the BAPI_USER_CREATE1 module.

Note however, when you use BAPI_USER_CREATE1 in a CUA landscape, you create a user in the central system. The user is initially inactive and cannot log on until roles or profiles are assigned. See BAPIs for Role Assignment [Page 16] or BAPIs for Profile Assignment [Page 17] and Considerations for Role and Profile Assignment Functions [Page 22].

1.5.4.2 Modifying a User

When modifying a user in a CUA landscape, take the following into consideration:

- The API for modifying a user is BAPI_USER_CHANGE as described in BAPI for Modifying Users [Page 11].
- When changing user attributes using this BAPI, you change the attributes in the central system.
- It is possible to change multiple attributes and the changes are executed according to the setting associated with each attribute. Therefore, global attributes are changed in the central system and distributed and those attributes that are to be maintained locally are filtered out and not changed.

Local attributes should be maintained using the maintenance functions (transactions SU01 or SU3) in the child systems.

1.5.4.3 Locking or Unlocking Users

The function modules to use for locking or unlocking users are BAPI_USER_LOCK and BAPI_USER_UNLOCK respectively.

There are two different types of locks that can be set in a CUA landscape: a global lock in the central system and local locks in child systems. When setting a lock when using an IAM system that is connected to the CUA central system, a global lock is set.

If both global and local locks are set, then unlocking a user in the child system does not unlock the user in the CUA, and therefore, the global lock remains set.

1.5.4.4 Deleting Users

Because additional data is associated with users, for example, change documents, you should consider deactivating them instead of deleting them. To deactivate a user, either lock it or specify a validity period.

No license fees are collected for deactivated users.

If it is necessary to delete users, use the function module BAPI_USER_DELETE. If the IAM system is connected to the central CUA system, then the user is deleted in the central system and in the child systems.

1.5.4.5 Setting Initial Passwords and Changing Passwords

You can use BAPI_USER_CHANGE to set a user's initial password in the CUA's central system. This initial password is distributed to the child systems when a user is created. However, you can only change existing passwords locally, you cannot change them in the central system.

See also BAPIs for Setting Passwords [Page 13].

1.5.5 Considerations for Role and Profile Assignment Functions

In a CUA landscape, roles or profiles can be assigned to users either in the child systems or in the central system. If the role or profile assignment takes place in the central system, the central system must have information about which roles or profiles exist in which systems. The actual roles or profiles do not need to exist in the central system.

To maintain a user's role assignment, use the function module BAPI_USER_LOCACTGROUPS_READ to read the existing assignment. Modify it, and reassign the changed roles using the module BAPI_USER_LOCACTGROUPS_ASSIGN. (To maintain profile assignments, use the function modules BAPI_USER_LOCPROFILES_READ and BAPI_USER_LOCPROFILES_ASSIGN accordingly.)

To delete role or profile assignments, use the function modules BAPI_USER_LOCACTGROUPS_DELETE and BAPI_USER_LOCPROFILES_DELETE respectively.

For the recommended landscape where the IAM system is connected to the CUA central system, we recommend setting the role or profile assignment maintenance attribute to *global*. With this configuration, the assignment is maintained in the central system and distributed to the child systems.

If the assignment maintenance attribute is set to *local*, then the role or profile is only assigned in the central system. The local administrators then have to maintain the role or profile assignments using the maintenance transactions SU01 or PFCG.

1.6 Appendix

The following topics provide summaries of the IAM BAPIs and function modules:

- Overview of IAM BAPIs and Function Modules [Page 24]
- Parameters for BAPI_HELPVALUES_GET [Page 25]
- Includes Used by IAM_API_TESTFRAME [Page 34]

1.6.1 Overview of IAM BAPIs and Function Modules

The table below summarizes the BAPIs and function modules used for identity management in SAP NetWeaver.

BAPIs and Function Modules Used for Identity Management

Purpose	RFC Function	BAPI Method
Obtain a list of users	BAPI_USER_GETLIST	*USER.GetList()*
Obtain information about users	BAPI_USER_GET_DETAIL	*USER.GetDetail()*
Value help	BAPI_HELPVALUES_GET	*Helpvalues.GetList()*
Create users	BAPI_USER_CREATE1	*USER.Create1()*
Modify users	BAPI_USER_CHANGE	*USER.Change()*
Delete users	BAPI_USER_DELETE	*USER.Delete()*
Set initial passwords	BAPI_USER_CHANGE	*USER.Change()*
Set a productive password	SUSR_USER_CHANGE_PASSWORD_RFC	None
Lock users	BAPI_USER_LOCK	*USER.Lock()*
Unlock users	BAPI_USER_UNLOCK	*USER.Unlock()*
Obtain a list of companies	BAPI_HELPVALUES_GET	*Helpvalues.GetList()*
Obtain information about a company	BAPI_ADDRESSORG_DETAIL	*AddressOrg.FindDetail()*
Modify address information of a company	BAPI_ADDRESSORG_CHANGE	*AddressOrg.Update()*
Obtain a list of roles	PRGN_ROLE_GETLIST	None
List role assignments	BAPI_USER_GET_DETAIL	*USER.GetDetail()*
Assign roles	BAPI_USER_ACTGROUPS_ASSIGN	*USER.ActgroupsAssign()*
Delete role assignments	BAPI_USER_ACTGROUPS_DELETE	*USER.ActgroupsDelete()*
Obtain a list of profiles	BAPI_HELPVALUES_GET	*Helpvalues.GetList()*
List profile assignments	BAPI_USER_GET_DETAIL	*USER.GetDetail()*
Assign profiles	BAPI_USER_PROFILES_ASSIGN	*USER.ProfilesAssign()*
Delete profile assignments	BAPI_USER_PROFILES_DELETE	*USER.ProfilesDelete()*

You can see how these functions are used in the test program IAM_API_TESTFRAME.

The table below summarizes the BAPIs to use for certain functions if the IAM system connects to a CUA landscape.

BAPIs Used for CUA Identity Management

Purpose	RFC Function	BAPI Method
Read (prior to modifying) role assignments	BAPI_USER_LOCACTGROUPS_READ	*USER.LocActgroupsRead*
Change role assignments	BAPI_USER_LOCACTGROUPS_ASSIGN	*USER.LocActgroupsAssign*
Delete role assignments	BAPI_USER_LOCACTGROUPS_DELETE	*USER.LocActgroupsDelete*
Read (prior to modifying) profile assignments	BAPI_USER_LOCPROFILES_READ	*USER.LocProfilesRead*
Change profile assignments	BAPI_USER_LOCPROFILES_ASSIGN	*USER.LocProfilesAssign*
Delete profile assignments	BAPI_USER_LOCPROFILES_DELETE	*USER.LocProfilesDelete*

1.6.2 Sample Parameters for BAPI_HELPVALUES_GET

The function module BAPI_HELPVALUES_GET can be used to obtain value help (F4-help) for several IAM-relevant object attributes, for example, roles, profiles or companies.

The tables below show possible uses of the module BAPI_HELPVALUES_GET to obtain such information.

You can use the function builder (transaction SE37) to test the function module using these parameters. Use the default values for parameters that are not provided in the tables, for example *OBJNAME*=<blank>, *FIELD*=<blank>, or *DESCRIPTIONONLY*=<blank>. Use *MAX_OF_ROWS* to limit the number of entries returned.

1.6.2.1 User Default Values

Value Help: Date Format

Parameter (Import)	Value
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	DEFAULTS
FIELD	DATFM
Table Parameter (Export)	**Value**
HELPVALUES	<list of available date formats>
VALUES_FOR_FIELD	<list of values that apply to the date formats>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions, for example, for list headings>

Example:

```
* Call the API
  call function 'BAPI_HELPVALUES_GET'
    exporting
      objtype                          = 'USER'
*   OBJNAME                          = ' '
      method                           = 'GETDETAIL'
      parameter                        = 'DEFAULTS'
      field                            = 'DATFM'
*   EXPLICIT_SHLP                    =
*   MAX_OF_ROWS                      = 0
*   DESCRIPTIONONLY                  = ' '
    importing
      return                           = lsreturn
    tables
*   SELECTION_FOR_HELPVALUES         =
      helpvalues                       = helpvalues
      values_for_field                 = values_for_field
      description_for_helpvalues       = description
            .
```

Value Help: Date Format (with Specific Search Help Type)

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	DEFAULTS
FIELD	DATFM
EXPLICIT_SHLP: SHLPNAME	XUDATFM
EXPLICIT_SHLP: SH	FV
Table Parameter (Export)	**Value**
HELPVALUES	<list of available date formats>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Decimal Point Formats (with Specific Search Help Type)

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	DEFAULTS
FIELD	DCPFM
EXPLICIT_SHLP: SHLPNAME	XUDCPFM
EXPLICIT_SHLP: SH	FV
Table Parameter (Export)	**Value**
HELPVALUES	<list of decimal point formats>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Printers (with Specific Search Help Type)

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	DEFAULTS
FIELD	SPLD
EXPLICIT_SHLP: SHLPNAME	H_TSP03
EXPLICIT_SHLP: SH	SH
Table Parameter (Export)	**Value**
HELPVALUES	<list of available printers>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Start Menus (with Specific Search Help Type)

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	DEFAULTS
FIELD	START_Menu
EXPLICIT_SHLP: SHLPNAME	S_TTREE_BMEN
EXPLICIT_SHLP: SH	SH
Table Parameter (Export)	**Value**
HELPVALUES	<list of available start menus>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

1.6.2.2 Company Address

Value Help: Company Address

Parameter (Import)	**Value**
OBJTYPE	USRCOMPANY
METHOD	DISPLAY
PARAMETER	COMPANY
Table Parameter (Export)	**Value**
HELPVALUES	<list of available company addresses>
VALUES_FOR_FIELD	<list of values to use for company addresses>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Company Address (with Specific Search Help Type)

Parameter (Import)	Value
OBJTYPE	USRCOMPANY
METHOD	DISPLAY
PARAMETER	COMPANY
EXPLICIT_SHLP: SHLPNAME	USCOMPANY_ADDR
EXPLICIT_SHLP: SH	SH
Table Parameter (Export)	**Value**
HELPVALUES	<list of available company addresses
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

1.6.2.3 Profiles

You cannot use the value help to obtain system-specific profiles from the CUA. You can only obtain profiles from the local systems.

Value Help: Single Profiles (with Specific Search Help Type)

Parameter (Import)	Value
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	PROFILES
EXPLICIT_SHLP: SHLPNAME	PROFILES_SINGLE_ACTIVE
EXPLICIT_SHLP: SH	SH
Table Parameter (Import)	**Value**
SELECTION_FOR_HELPVALUES: SELECT_FLD	TYP
SELECTION_FOR_HELPVALUES: SIGN	I
SELECTION_FOR_HELPVALUES: OPTION	EQ
SELECTION_FOR_HELPVALUES: LOW	S
SELECTION_FOR_HELPVALUES: HIGH	<blank>
Table Parameter (Export)	**Value**
HELPVALUES	<list of available single profiles>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Composite Profiles (with Specific Search Help Type)

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	PROFILES
FIELD	<blank>
EXPLICIT_SHLP: SHLPNAME	PROFILES_COMPOSITE_ACTIVE
EXPLICIT_SHLP: SH	SH
Table Parameter (Import)	**Value**
SELECTION_FOR_HELPVALUES: SELECT_FLD	TYP
SELECTION_FOR_HELPVALUES: SIGN	I
SELECTION_FOR_HELPVALUES: OPTION	EQ
SELECTION_FOR_HELPVALUES: LOW	C
SELECTION_FOR_HELPVALUES: HIGH	<blank>
Table Parameter (Export)	**Value**
HELPVALUES	<list of available composite profiles>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Generated Profiles (with Specific Search Help Type)

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	PROFILES
EXPLICIT_SHLP: SHLPNAME	PROFILES_GENERATED_ACTIVE
EXPLICIT_SHLP: SH	SH
Table Parameter (Import)	**Value**
SELECTION_FOR_HELPVALUES: SELECT_FLD	TYP
SELECTION_FOR_HELPVALUES: SIGN	I
SELECTION_FOR_HELPVALUES: OPTION	EQ
SELECTION_FOR_HELPVALUES: LOW	G

Table Parameter (Import)	Value
SELECTION_FOR_HELPVALUES: HIGH	<blank>
Table Parameter (Export)	**Value**
HELPVALUES	<list of available generated profiles>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: All Profiles (with Specific Search Help Type)

Parameter (Import)	Value
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	PROFILES
EXPLICIT_SHLP: SHLPNAME	PROFILES_SINGLE_ACTIVE
EXPLICIT_SHLP:SH	SH
Table Parameter (Export)	**Value**
HELPVALUES	<list of all available profiles>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

1.6.2.4 Roles

You cannot use the value help to obtain system-specific roles from the CUA. You can only obtain the roles from the local systems.

Value Help: Single Roles (with Specific Search Help Type)

Parameter (Import)	Value
OBJTYPE	USER
METHOD	ACTGROUPSASSIGN
PARAMETER	ACTIVITYGROUPS
FIELD	AGR_NAME
EXPLICIT_SHLP: SHLPNAME	AGR_SINGLE
EXPLICIT_SHLP: SH	SH
Table Parameter (Export)	**Value**
HELPVALUES	<list of available single roles>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Composite Roles (with Specific Search Help Type)

Parameter (Import)	Value
OBJTYPE	USER
METHOD	ACTGROUPSASSIGN
PARAMETER	ACTIVITYGROUPS
FIELD	AGR_NAME
EXPLICIT_SHLP: SHLPNAME	AGR_COLL
EXPLICIT_SHLP: SH	SH
Table Parameter (Export)	**Value**
HELPVALUES	<list of available composite roles>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

1.6.2.5 Users

Although you can use the value help BAPI to obtain a list of users, alternatively, you can use BAPI_USER_GETLIST. (See BAPIs for User Enquiry [Page 8].)

Value Help: User by Last Name (with Specific Search Help Type)

Parameter (Import)	Value
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	USERNAME
EXPLICIT_SHLP: SHLPNAME	USER_ADDR
EXPLICIT_SHLP:SH	SH
Table Parameter (Import)	**Value**
SELECTION_FOR_HELPVALUES: SELECT_FLD	MC_NAMELAS
SELECTION_FOR_HELPVALUES: SIGN	I
SELECTION_FOR_HELPVALUES: OPTION	EQ
SELECTION_FOR_HELPVALUES: LOW	<last_name_of_user>
SELECTION_FOR_HELPVALUES: HIGH	<blank>
Table Parameter (Export)	**Value**
HELPVALUES	<user accounts whose last name matches search criteria>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: User by User Group

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	USERNAME
Table Parameter (Import)	**Value**
SELECTION_FOR_HELPVALUES: SELECT_FLD	CLASS
SELECTION_FOR_HELPVALUES: SIGN	I
SELECTION_FOR_HELPVALUES: OPTION	CP
SELECTION_FOR_HELPVALUES: LOW	L*
SELECTION_FOR_HELPVALUES: HIGH	<blank>
Table Parameter (Export)	**Value**
HELPVALUES	<user accounts whose user group matches search criteria>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

1.6.2.6 Reference Users

Value Help: Reference User

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	USERNAME
Table Parameter (Import)	**Value**
SELECTION_FOR_HELPVALUES: SELECT_FLD	USTYP
SELECTION_FOR_HELPVALUES: SIGN	I
SELECTION_FOR_HELPVALUES: OPTION	EQ
SELECTION_FOR_HELPVALUES: LOW	L
SELECTION_FOR_HELPVALUES: HIGH	<blank>

Table Parameter (Export)	Value
HELPVALUES	<list of available reference users>
VALUES_FOR_FIELD	<list of values that apply to the reference users>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

Value Help: Reference User (with Specific Search Help Type)

Parameter (Import)	Value
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	USERNAME
EXPLICIT_SHLP: SHLPNAME	USREFUSER
EXPLICIT_SHLP: SH	SH
Table Parameter (Import)	**Value**
SELECTION_FOR_HELPVALUES: SELECT_FLD	USTYP
SELECTION_FOR_HELPVALUES: SIGN	I
SELECTION_FOR_HELPVALUES: OPTION	EQ
SELECTION_FOR_HELPVALUES: LOW	L
SELECTION_FOR_HELPVALUES: HIGH	<blank>
Table Parameter (Export)	**Value**
HELPVALUES	<list of available reference users>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

1.6.2.7 User Groups

Value Help: User Groups (available as of SAP NW 2004s)

Parameter (Import)	**Value**
OBJTYPE	USER
METHOD	GETDETAIL
PARAMETER	LOGONDATA
FIELD	CLASS
EXPLICIT_SHLP: SHLPNAME	USGRP
EXPLICIT_SHLP: SH	CT
Table Parameter (Export)	**Value**
HELPVALUES	<list of available user groups>
DESCRIPTION_FOR_HELPVALUES	<metadata to use for descriptions>

1.6.3 Includes Used by IAM_API_TESTFRAME

The table below summarizes the functions demonstrated by the test program IAM_API_TESTFRAME.

IAM_API_TESTFRAME Functions

Function	**Include**	**BAPI or Function Module Used**
Get parameters for the input screen	IAM_TOP	None
Obtain a list of users	IAM_USERLIST	BAPI_USER_GETLIST
Obtain information about users	IAM_USERDETAILS	BAPI_USER_GET_DETAIL
Create users	IAM_USERCREATE	BAPI_USER_CREATE1
Modify users (also specifies a second input screen)	IAM_USERCHANGE	BAPI_USER_CHANGE
Lock users	IAM_USERLOCK	BAPI_USER_LOCK
Unlock users	IAM_USERLOCK	BAPI_USER_UNLOCK
List available companies	IAM_COMPANYLIST	BAPI_HELPVALUES_GET
Obtain information about a company	IAM_COMPANY DETAILS	BAPI_ADDRESSORG_ GETDETAIL
List available roles	IAM_ROLELIST	PRGN_GET_ROLELIST
List role assignments	IAM_USEROLES	BAPI_USER_GET_DETAIL
Assign roles to users	IAM_ROLEASSIGN	BAPI_USER_ACTGROUPS_ ASSIGN
Delete role assignments	IAM_ROLEASSIGN_ DELETE	BAPI_USER_ACTGROUPS_ DELETE
List available profiles	IAM_PROFILELIST	BAPI_HELPVALUES_GET

2 Identity Management Java APIs

2.1 Purpose

This section describes how to use the identity management Java APIs for managing users, groups, and roles when using the User Management Engine (UME) for identity management with SAP systems. When using this scenario, the available UME APIs are provided using the Service Provisioning Markup Language (SPML) standard.

For more information about SPML, see **`www.openspml.org`** or **`www.oasis-open.org`**.

This documentation does not describe the details of each API as these are documented in the JavaDocs for the API; it simply introduces the concepts and highlights those aspects that need special consideration.

2.2 Integration

The AS Java accepts and processes the SPML request using Simple Object Access Protocol (SOAP) messages (according to the SPML 1.0 Bindings specification).

The URL address used by SPML service on the AS Java is **`<server>:<port>/spml/provisioning`**.

The previously used URL address (**`<server>:<port>/spml/spmlservice`**) is still valid.

Note the following:

- When using the new URL, you have to use SPML-specific object IDs instead of the UME unique IDs. For more information, see SPML-Specific Object IDs [Page 37].
- When using the new URL, the logon ID for users and the `uniquename` attribute for roles and groups must be unique.
- For inbound requests, both ID patterns are allowed. For outbound responses, you will receive the ID pattern that corresponds to the URL used.

2.3 Features

- You can perform the following functions on user, group and role objects using the identity management SPML APIs:
  - Creating objects
  - Modifying objects
  - Searching for objects
  - Deleting objects

  These functions can also be bundled together in batch requests.
- The APIs can be used for user management with the UME with all of the supported data sources (SAP system database, LDAP server or other database).

2.4 Constraints

- SAP role objects cannot be created or deleted using these APIs.
- The use of digital certificates is not supported.
- Only certain ABAP attributes are supported.

2.5 Prerequisites

Available Releases

The APIs are available as of SAP NetWeaver ’04 SPS 14 and SAP NetWeaver 7.0 SPS 05.

Security Role Assignments

Assign the UME actions *UME.Spml_Read_Action* and *UME.Spml_Write_Action* to control access to the SPML service. Users who are assigned the action *UME.Spml_Write_Action* (or *UME.Manage_All*) are allowed to use the complete function set belonging to the SPML service. Users who are assigned the UME action *UME.Spml_Read_Action* are only allowed to search and read the SPML schema.

2.6 APIs for User Administration Functions in Java

SPML APIs are available for use with the UME for the following functions:

2.6.1 SPML-Specific Object IDs

The SPML-specific object IDs use the following patterns for roles, groups, and users.

- Roles and Groups: `SPML.<objectclass>.<uniquename>`
- Users: `SPML.<objectclass>.<logonname>`

Use these identifiers in the API calls when using the provisioning URL (/spml/provisioning) instead of the SPML service URL (/spml/spmlservice). When using the SPML service URL, use the UME unique IDs.

2.6.1.1 Example

The following examples show a modify request using the SPML-specific object ID and the UME unique ID, respectively.

SPML-Specific Object IDs

```
<modifyRequest>
      <identifier type="GenericString">
            <id>SPML.SAPGROUP.SAPTestGroup_1</id>
      </identifier>
      <modifications>
            <modification name="member" operation="add">
                  <value>SPML.SAPUSER.Administrator</value>
            </modification>
      </modifications>
</modifyRequest>
```

UME Unique IDs

```
<modifyRequest>
      <identifier type="GenericString">
            <id>GRUP.PRIVATE_DATASOURCE.un:SAPTestGroup_1</id>
      </identifier>
      <modifications>
            <modification name="member" operation="add">

      <value>USER.PRIVATE_DATASOURCE.un:Administrator</value>
            </modification>
      </modifications>
</modifyRequest>
```

2.6.2 Reading the Schema

2.6.2.1 Use

The schema contains the description, object class names and attribute names defined in the UME SPML API. Before you perform any functions using the UME SPML API, you need to read the schema to obtain the available attributes.

The default schema provided with SAP NetWeaver 7.0 is provided with the AS Java. You can find a description of the attributes in the Appendix [Page 50].

2.6.2.2 Syntax

The SPML request to read the schema is:

```
<schemaRequest requestID="schema_01">
    <schemaIdentifier
            schemaIDType="urn:oasis:names:tc:SPML:1:0#GenericString">
        <schemaID>SAPprincipals</schemaID>
    </schemaIdentifier>
</schemaRequest>
```

The SPML response contains the schema defined by the UME SPML API.

2.6.3 Creating Objects

2.6.3.1 Use

Use the SPML request `addRequest` to create objects defined in the schema, that is `sapuser` and `sapgroup` objects. The SPML service on the AS Java creates and returns the object's ID.

You cannot create roles using the SPML create request. Create roles in the backend system.

2.6.3.2 Example

The following examples show how to use the SPML request for creating objects.

SPML Request for Creating a User

```
    <spml:addRequest requestID="add-1"
          xmlns="urn:oasis:names:tc:SPML:1:0"
      xmlns:spml="urn:oasis:names:tc:SPML:1:0"
      xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">

       <spml:attributes>
          <spml:attr name="objectclass"
                    xmlns="urn:oasis:names:tc:DSML:2:0:core">
                <dsml:value>sapuser</dsml:value>
          </spml:attr>
          <spml:attr name="logonname"
                    xmlns="urn:oasis:names:tc:DSML:2:0:core">
             <dsml:value>spmltest</dsml:value>
          </spml:attr>
          <spml:attr name="lastname"
                    xmlns="urn:oasis:names:tc:DSML:2:0:core">
             <dsml:value>Test</dsml:value>
          </spml:attr>
          <spml:attr name="firstname"
                    xmlns="urn:oasis:names:tc:DSML:2:0:core">
              <dsml:value>Hugo</dsml:value>
          </spml:attr>
          <spml:attr name="password"
                    xmlns="urn:oasis:names:tc:DSML:2:0:core">
              <dsml:value>initial01</dsml:value>
          </spml:attr>
           <spml:attr name="validto"
                    xmlns="urn:oasis:names:tc:DSML:2:0:core">
              <dsml:value>20051031000000Z</dsml:value>
           </spml:attr>
       </spml:attributes>
    </spml:addRequest>
```

SPML Response After Creating a User

```
    <addResponse xmlns="urn:oasis:names:tc:SPML:1:0"
result="urn:oasis:names:tc:SPML:1:0#success" requestID="add-1">
      <identifier xmlns=""
             type="urn:oasis:names:tc:SPML:1:0#GenericString">
                 <id>SPML.SAPUSER.spmltest</id>
      </identifier>
  </addResponse>
```

SPML Request for Creating a Group

This request creates the group with the unique name SAPTestGroup_1.

```
      <addRequest requestID="create_1">
            <attributes>
                  <attr name="objectclass">
                        <value>sapgroup</value>
                  </attr>
                  <attr name="uniquename">
                        <value>SAPTestGroup_1</value>
                  </attr>
                  <attr name="description">
                        <value>test group</value>
                  </attr>
            </attributes>
      </addRequest>
```

SPML Request for Creating a User and Assigning a Role During Creation

```
<spml:addRequest requestID="add-1"
          xmlns="urn:oasis:names:tc:SPML:1:0"
      xmlns:spml="urn:oasis:names:tc:SPML:1:0"
      xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">

       <spml:attributes>
          <spml:attr name="objectclass">
                <dsml:value>sapuser</dsml:value>
          </spml:attr>
          <spml:attr name="logonname">
             <dsml:value>IDM_TESTUSER</dsml:value>
          </spml:attr>
          <spml:attr name="lastname">
             <dsml:value>Test</dsml:value>
          </spml:attr>
          <spml:attr name="firstname">
              <dsml:value>Hugo</dsml:value>
          </spml:attr>
           <spml:attr name="validto">
              <dsml:value>20091031000000Z</dsml:value>
           </spml:attr>
           <spml:attr name="password">
                  <dsml:value>init001</dsml:value>
            </spml:attr>
            <spml:attr name="passwordchangerequired">
                  <dsml:value>false</dsml:value>
            </spml:attr>
            <spml:attr name="securitypolicy">
                  <dsml:value>technical</dsml:value>
            </spml:attr>
            <spml:attr name="allassignedroles">
                  <dsml:value>SPML.SAPROLE.Role3</dsml:value>
            </spml:attr>
            <spml:attr name="assignedroles">
```

2.6.4 Modifying Objects

2.6.4.1 Use

Use the SPML request `modifyRequest` to modify `sapuser` and `sapgroup` objects that available using the UME SPML API.

2.6.4.2 Prerequisites

You know the object's ID.

To obtain the object's user ID, use the `searchRequest` request to search for the object and obtain its ID. For more information, see Searching for Objects or Obtaining Attribute Values for Objects [Page 46].

2.6.4.3 Example

The following examples show how to use the SPML request for modifying objects. Insert the object's ID in the `<id>` tag in the `<identifier>` block.

SPML Request for Changing or Adding New Attributes

```
<modifyRequest
      requestID="mod_041104_3">
      <identifier
            type="GenericString">
            <id>SPML.SAPUSER.spmltest</id>
      </identifier>
      <modifications>
            <modification
                  name="islocked">
                  <value>true</value>
            </modification>
            <modification
                  name="validto">
                  <value>20061231000000Z</value>
            </modification>
            <modification
                  name="lastname">
                  <value>Test Last Name</value>
            </modification>
            <modification
                  name="email">
                  <value>spml.test@mycompany.org</value>
            </modification>
      </modifications>
</modifyRequest>
```

SPML Response for Modifying Objects

This response indicates that the changes were processed successfully.

```
<modifyResponse xmlns="urn:oasis:names:tc:SPML:1:0"
requestID="mod_041104_3"
result="urn:oasis:names:tc:SPML:1:0#success"/>
```

SPML Request for Assigning a User to a New Group

This request assigns the user `Administrator` to the group `SAPTestGroup_1`.

```
<modifyRequest>
      <identifier
            type="GenericString">
            <id>SPML.SAPGROUP.SAPTestGroup_1</id>
      </identifier>
      <modifications>
            <modification
                  name="member"
                  operation="add">
            <value>SPML.SAPUSER.Administrator</value>
            </modification>
      </modifications>
</modifyRequest>
```

SPML Request for Assigning a User to a Role

This request assigns the user `Administrator` to the role `TestAdmins`.

```
<modifyRequest>
      <identifier
            type="GenericString">
            <id>SPML.SAPROLE.TestAdmins</id>
      </identifier>
      <modifications>
            <modification
                  name="member"
                  operation="add">
            <value>SPML.SAPUSER.Administrator</value>
            </modification>
      </modifications>
</modifyRequest>
```

SPML Request for Modifying a User: Deleting all Role Assignments and Adding a New Group Assignment)

This request deletes all roles from the role assignments for the user TESTUSER and adds the group TESTGROUP.

```
   <spml:modifyRequest requestID="mod-1"
      xmlns:spml="urn:oasis:names:tc:SPML:1:0"
      xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
      >
      <spml:identifier
            type="urn:oasis:names:tc:SPML:1:0#GenericString">
            <spml:id>SPML.SAPUSER.TESTUSER</spml:id>
      </spml:identifier>
      <spml:modifications>
            <spml:modification name="lastname" operation="replace">
                  <dsml:value>Test</dsml:value>
            </spml:modification>
            <spml:modification name="assignedroles"
operation="delete">
            </spml:modification>

            <spml:modification name="assignedgroups" operation="add">
                  <dsml:value>SPML.SAPGROUP.TESTGROUP</dsml:value>
      </spml:modification>
      </spml:modifications>
   </spml:modifyRequest>
```

2.6.5 Deleting Objects

2.6.5.1 Use

Use the SPML request `deleteRequest` to delete a single object.

2.6.5.2 Prerequisites

The object's unique ID is known.

To obtain the object's user ID, use the `searchRequest` request to search for the object and obtain its ID. For more information, see Searching for Objects or Obtaining Attribute Values for Objects [Page 46].

2.6.5.3 Example

The following examples show how to use the SPML request for deleting objects.

SPML Request for Deleting an Object

```
<deleteRequest
      requestID="del_1">
      <identifier
            type="GenericString">
            <id>SPML.SAPGROUP.SAPTestGroup_1</id>
      </identifier>
</deleteRequest>
```

SPML Response After Deleting an Object

```
<deleteResponse xmlns="urn:oasis:names:tc:SPML:1:0"
result="urn:oasis:names:tc:SPML:1:0#success" requestID="del_1"/>
```

2.6.6 Changing or Resetting Passwords

2.6.6.1 Use

Use the modify request as described in Modifying Objects [Page 41] to change or reset passwords. Note the following:

- To change a user's password, include both the old and new passwords in the modification element in the request. The corresponding attributes are `oldpassword` and `password`.
- To reset a user's password, only include the new password in the password attribute in the modification request. In this case, the password has an initial status and must be changed the next time the user logs on.

2.6.6.2 Example

SPML Request for Changing a User's Password

```
<modifyRequest
      requestID="mod_041104_3">
      <identifier
            type="GenericString">
            <id>SPML.SAPUSER.spmluser</id>
      </identifier>
      <modifications>
            <modification
                  name="oldpassword">
                  <value>password</value>
            </modification>
            <modification
                  name="password">
                  <value>newpassword</value>
            </modification>
      </modifications>
</modifyRequest>
```

SPML Request for Resetting a User's Password

```
<modifyRequest
      requestID="mod_041104_3">
      <identifier
            type="GenericString">
            <id>SPML.SAPUSER.spmluser</id>
      </identifier>
      <modifications>
            <modification
                  name="password">
                  <value>newpassword</value>
            </modification>
      </modifications>
</modifyRequest>
```

2.6.7 Locking and Unlocking Users

2.6.7.1 Use

To lock or unlock a user, use the modify request as described in Modifying Objects [Page 41]. To lock the user set the `islocked` attribute to `true`. To unlock the user, set the attribute to `false`.

2.6.7.2 Example

SPML Request for Locking a User

```
<modifyRequest
      requestID="mod_041104_3">
      <identifier
            type="GenericString">
            <id>SPML.SAPUSER.spmluser</id>
      </identifier>
      <modifications>
            <modification
                  name="islocked">
                  <value>true</value>
            </modification>
      </modifications>
</modifyRequest>
```

2.6.8 Searching for Objects or Obtaining Attribute Values for Objects

2.6.8.1 Use

Use the SPML request `searchRequest` to search for objects defined in the schema. When sending a search request, you also specify the attributes that should be returned for the object. In this way, you can also obtain attribute values for specific objects.

The search request consists of three elements:

- `<searchBase>`: Specifies the starting point for the search
- `<filter>`: Specifies the filter to use for searching
- `<attributes>`: Specifies the attributes to return.

2.6.8.1.1 Search Filters

The filter contains a set of criteria to search for using either the `<equalityMatch>` element for a complete match, or the `<substrings>` element for a match containing the substring.

Place the filter elements in a conditional operator block using `<and>` or `<or>` elements to specify how the elements are to be considered.

2.6.8.1.2 Object Classes

When searching for objects, you first need to specify the object class to search for. The object classes specified in the schema provided with the AS Java are `sapuser`, `sapgroup`, and `saprole`. To specify which class to use, you can either:

- Specify the object class as an ID in the search base as shown below

```
<searchBase
  type="urn:urn:oasis:names:tc:SPML:1:0#GenericString">
  <id>sapuser</id>
</searchBase>
```

- Specify the object class in the search filter as shown below

```
<filter>
  <and>
        <equalityMatch
              name="objectclass">
              <value>sapuser</value>
        </equalityMatch>
        <substrings
              name="logonname">
              <initial>d</initial>
        </substrings>
  </and>
</filter>
```

If you specify the object class within the filter, then set up your filter to search for the object class and additional filter elements by including an <and> conditional block in the filter's first level.

For additional filter elements, only one level of conditions is supported, using either <and> or <or>. You cannot use additional nested conditions, nor can you mix <and> and <or> conditional operators.

2.6.8.1.3 Obtaining Attributes for an Object

Use the <attributes> element to specify which attributes should be returned by the request.

In this way, you can retrieve attribute values for objects, for example, to obtain the object's ID, which is needed for further operations such as modifying or deleting objects.

2.6.8.2 Example

For examples of search requests and responses, see Examples for Search Requests and Responses [SAP Library].

2.6.9 Using Batch Functions

2.6.9.1 Use

Use the SPML request batchRequest to consolidate user management functions and process them in batch mode.

The application calling the batch request has to set a unique request ID. This ID is used to obtain the batch process's status. Therefore, if the request ID set by the application is not unique, then the batch request will fail.

Single requests are always processed synchronously. Batch requests can be processed synchronously or asynchronously. For synchronous requests, you can also specify that the single requests are to be processed sequentially or in parallel. See the table below for an overview of the possible processing methods.

Batch Processing Methods

Timing Method	Queuing Method
Synchronous	Sequential or parallel
Asynchronous	Parallel

Note the following:

- If you specify a batch request to be processed asynchronously as well as sequentially, it will be processed synchronously.
- The results of the batch request can only be read using the status request where the request ID corresponds to the ID used for the original batch request.
- Unless the batch processing type is synchronous and sequential, the batch response contains the information that the batch request is pending.
- Batch processing results are available for a limited period of time only (one day).

2.6.9.2 Example

The following example shows how to use the SPML batch request for managing objects.

SPML Batch Request

This request creates the group `SAPTestGroup_2` and the users `SAPUser1` and `SAPUser2`.

```
<batchRequest requestID="b2">
<addRequest requestID="create_1">
<attributes>
<attr name="objectclass">
<value>sapgroup</value>
</attr>
<attr name="uniquename">
<value>SAPTestGroup_2</value>
</attr>
<attr name="description">
<value>test group</value>
</attr>
</attributes>
</addRequest>
<addRequest requestID="create_2">
<attributes>
<attr name="objectclass">
<value>sapuser</value>
</attr>
<attr name="logonname">
<value>SapUser1</value>
</attr>
<attr name="lastname">
<value>User1</value>
</attr>
<attr name="firstname">
<value>SAP</value>
</attr>
</attributes>
</addRequest>
<addRequest requestID="create_3">
<attributes>
<attr name="objectclass">
<value>sapuser</value>
</attr>
<attr name="logonname">
<value>SapUser2</value>
</attr>
```

```
<attr name="lastname">
<value>User2</value>
</attr>
<attr name="firstname">
<value>SAP</value>
</attr>
</attributes>
</addRequest>
</batchRequest>
```

SPML Batch Status Request

The following example shows how to obtain the status of the batch request with the ID b2.

```
<statusRequest requestID="b2"/>
```

SPML Cancel Batch Request

The following example shows how to cancel the batch quest with the ID b2.

```
<cancelRequest requestID="b2"/>
```

2.7 Appendix: Schema Description

The tables below provide an overview of the schema description that is provided with the AS Java and used by the identity management APIs. You can read the schema description using the SPML request `schemaRequest`.

The schema description is subject to change in future releases. Therefore, for the most current and complete description, see the `schema.xml` file that is provided with the AS Java.

Schema Identifiers

Identifier	Value
`providerID`	SAP
`schemaID`	SAPprincipals

Object Classes

Object Class	Description
`sapuser`	SAP system user object
`saprole`	SAP system role object
`sapgroup`	SAP system group object

Attributes Used by Object Classes

Attribute	Used by sapuser	Used by saprole	Used by sapgroup	Comment
`logonname`	X			Required attribute when creating users.
`firstname`	X			
`lastname`	X			
`salutation`	X			
`title`	X			
`jobtitle`	X			
`mobile`	X			
`telephone`	X			
`displayname`	X	X	X	
`description`	X	X	X	
`password`	X			See Changing or Resetting Passwords [Page 44].

Attribute	Used by sapuser	Used by saprole	Used by sapgroup	Comment
`oldpassword`	X			Necessary when changing passwords. See Changing or Resetting Passwords [Page 44].
`email`	X			
`fax`	X			
`locale`	X			
`timezone`	X			
`validfrom`	X			
`validto`	X			
`certificate`	X			
`lastmodifydate`	X	X	X	
`islocked`	X			Boolean value (`true` or `false`). See also Locking and Unlocking Users [Page 45].
`uniquename`		X	X	
`member`		X	X	Specifies the users assigned to either groups or roles.
`department`	X			
`id`	X	X	X	Read-only. Set by the AS Java when an object is created.
`isserviceuser`	X			Defines if a user is a service user or a normal user

Attribute	Used by sapuser	Used by saprole	Used by sapgroup	Comment
`securitypolicy`	X			Specifies the type of the enabled security policies of the user (default, technical, unknown). (Only relevant for Release 7.0 and higher.)
`datasource`	X	X	X	Read-only. Specifies the home data source of the object.
`assignedroles`	X			List of all directly assigned roles.
`allassignedroles`	X			Read-only. List of all assigned roles.
`assignedgroups`	X			List of all directly assigned groups.
`allassignedgroups`	X			Read-only. List of all assigned groups.
`distinguishedname`	X			Returns the LDAP Distinguished Name if the object is stored on an LDAP server.

The attribute formats that are specified by the schema are primarily strings. However, additional formatting rules may apply according to the data source used, for example, for date formats.