Central Address Management (BC-SRV-ADR)

Release 4.6C

Copyright

Icons

Icon	Meaning
⚠	Caution
	Example
➡	Note
	Recommendation
	Syntax
💡	Tip

Contents

Central Address Management (BC-SRV-ADR)

Purpose

Central address management (CAM) offers functions for managing addresses in applications. Using various address types, you can store and further process addresses. An address is subordinate to an associated application object (such as a customer, a purchase order or a plant).

Implementation Considerations

For standard functions such as creating, changing, displaying and finding addresses, CAM provides flexible dialog integration. CAM therefore helps users find their way through different applications when they maintain addresses.

CAM function modules enable the application to store the addresses for the associated application object in CAM tables. These tables contain addresses of all applications that make use of CAM in a system. Functions and data storage in address management are central with regard to a client in a system.

Integration

CAM is already used by a broad range of applications, including, for example:

- Customer and vendor master
- Site master (Retail)
- Bank addresses
- Sales and Distribution documents: document addresses and one-time customer addresses
- User addresses
- SAPoffice addresses (external communication)
- Customizing addresses (sm30)
- MM Purchasing: permanent delivery addresses, manual delivery addresses in purchase orders, purchase requisitions, one-time vendors
- PM/SM (functional locations, equipment, notification, order)
- IS-U (connection objects, contracts)
- IS-Oil (service stations)
- SAP Business Partners
- New Dimension products such as SFA, BW, APO, CRM

In addition, CAM provides functions to support other tools and interfaces.

- Integration into *SAPconnect*
- Providing *archiving classes* for the Archive Development Kit (ADK)
- Supporting *addresses as objects* in the BOR
- Providing *interfaces for application-specific enhancements*

- Automatically writing change documents for address data changes

Features

General Characteristics

- Together with the address, you can enter data for all common communication types (for example, telephone number, fax, email, and so on).
- International address requirements (print output according to international mail standards, multiple address formats, for example, for Asian countries) are considered.
- By means of a where-used list you can determine which application objects reference an address as an attribute.
- Checks against city and street directories are performed through interfaces (for example, for partner solutions) or as a function in the standard system (city and street files are delivered without content and can be filled by means of transfer programs).
- Duplicate check and error-tolerant search through interfaces for third-party vendors that can be validated
- CAM provides functions for using addresses in combination with other tools (see *Integration*).

Integration into the Application

- Three address types are available to map addresses of the application onto CAM data structures: company addresses, personal addresses and workplace addresses
- Dialogs for maintaining addresses can be integrated flexibly into the application (as a dialog box, subscreen or full screen). You can parametrize the screens to hide fields or define fields as required, for example.
- Address data is updated together with the application data.
- Addresses are grouped by their assignment to an application which can be used as a filter for searches and authorization checks.

Address Distribution and Transport

Address used for	**Example**	**Transport or distribution methods or tools used by CAM**
Master data	Customer, vendor	Distribution through ALE (Application Link Enabling)
Customizing objects	Plant, sales organization	Methods for transporting addresses within the normal Customizing object transport process
System user	System user	Methods for client copy and BAPIs for central user management (cross-system in this case)

Constraints

The functions of central address management are not cross-system. However, you can use the tools and methods mentioned above to transport and distribute addresses.

Central Address Management (BC-SRV-ADR)

Time-dependent addresses (that are valid for a limited period only, such as the vacation address of a newspaper subscriber) are currently not supported.

Implementation Considerations

Advantages of Central Address Management

Addresses play an important role in a large number of business processes. In Accounting, for example, addresses are part of the customer master data. The addresses are collected and then used in follow-up processes such as sales orders. Further examples include address data of banks, vendors, and contacts. There are many areas in which it is natural to enter, change or use addresses for correspondence or other types of communication (email, telephone, fax).

Considering these aspects, a central organization of addresses in components of the SAP System provides the following advantages:

- Standardized interfaces can be used centrally by all components. This makes address maintenance easier.
- Consistent encapsulation of the functions and data reduces the maintenance effort in the components that use central address management.
- New functions related to address management can be implemented on a central basis. This is especially important with regard to globalization (for example, mail addresses must meet international requirements).

The SAP System has been benefiting from the advantages of central address management (CAM) since Release 4.0.

Contents of this Documentation

This documentation is primarily designed for employees in the areas of Customizing and development.

The section on the Basics [Page 10] describes important terms and concepts of CAM. You are required to make yourself familiar with these terms and concepts in order to understand the other sections.

The section on Working With CAM [Page 23] is a guide for developers who want to use CAM in an application component.

All other sections are not designed for a specific target group.

Basics

This section explains the basic terms and concepts of central address management.

Mapping of Application Addresses

Depending on the addressee of an address, an address is made up of different components. For a contact in a company, you need, for example, not only the company address, but also the name of the contact and possibly the name of the department. This is why it makes sense to distinguish between different types of addresses.

CAM provides three *address types*:

- Company addresses (address type 1)
- Personal addresses (address type 2)
- Workplace addresses (address type 3)

In case of address types 2 and 3, persons are assigned to the address data.

Address Type 1: Company Addresses

Short description	Example
Addresses of companies, plants, subsidiaries, and so on, can be mapped to this address type.	SAP America Inc. 3999 West Chester Pike Newtown Square, PA 19073 USA

An address of this type can be created regardless of an application object.

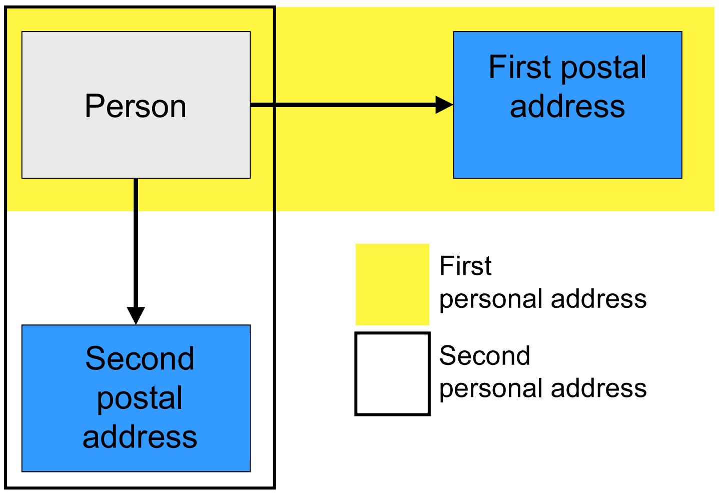
All addresses that you create in Customizing (definition of the organizational structure) are of this type.

In the customer master, the first address belongs to this address type.

Address Type 2: Personal Addresses

Short description	Example
Private address of a person. The address consists of address-relevant personal data and the postal address.	Mr. John Smith 100 Main Street Anytown N.Y. 12345

The person represents a separate object within central address management. A person can have one or more postal addresses (see figure):



Normally, you always maintain one postal address for a person (there are no standard dialogs available to create a person without an address.)

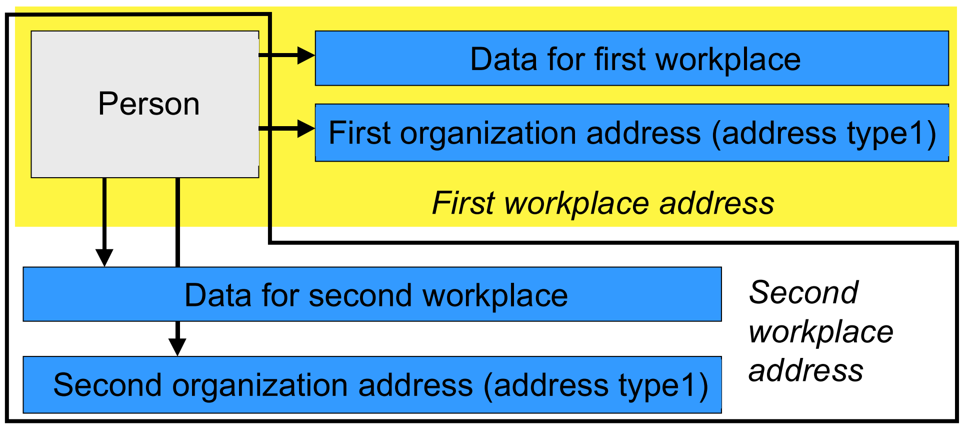
An address is assigned to exactly one person. No two persons can have a link to the same address. For two persons with the same address, you must create two personal addresses. The personal address as a whole (person and postal address), however, can be used more than once.

The section on Preparing to Work With CAM [Page 28] provides more details on the relations between application objects and addresses.

Address Type 3: Workplace Addresses

Short description	**Example**
Company-specific address of a person. The address consists of address-relevant personal data, workplace data (department, room number, extension, and so on) and the company address.	James Smith Sales Manager ABC Software 17 Charles Ave Los Angeles CA 06312

This address type consists of three subobjects: the person, the workplace data and the company address (address type 1):


Person
Data for first workplace
First organization address (address type1)
First workplace address
Data for second workplace
Second organization address (address type1)
Second workplace address

As with address type 2, a person can have more than one workplace.

For this address type, you cannot have a link to a personal address (address type 2) instead of to the company address, that is, combine address type 3 with address type 2. Instead, you must create a new personal address for that person.

You can create an address of this type without reference to an application object.

⚠

If the term *address* is used in the following, the statements made about this object are also applicable to the *person* object unless not specified otherwise.

Comparison of the Address Types

In contrast to the other two address types, adresses of address type 1 (company address) are not linked to a person.

Addresses of address types 2 and 3 are linked to a person. From a technical point of view, the following differences exist betweeen the address data:

- A workplace address (type 3) **points to** a company address (type 1) that already exists.
- The address data of personal addresses (type 2) constitute **attributes** of exactly one person. In the CAM data model, these attributes are not independent and can therefore not be referenced more than once (that is, by multiple persons).

These characteristics affect the maintenance of where-used lists, for example (see The Where-Used List [Page 18] and Maintaining Where-Used Lists [Page 37]).

Access to CAM Addresses

Working with the Buffer

Normally, addresses are part of an application object. To keep data consistent, you must update the address data together with the data of the application object. This is made possible by means of a buffer that manages all addresses on the application server until they are updated in the database. This buffer is available through function group `SZA0` and is referred to as the *local memory*.

Identifying Addresses

Using Handles to Access the Buffer

Applications use function modules to access the addresses of CAM. As long as an application does not save an address to the database, the application refers to an *address handle*. This handle identifies an address in the local memory that is currently being entered (for example, on a subscreen in the application program). The application developer is responsible for assigning these address handles.

See also: Creating an Address [Page 32].

Using Address Numbers to Access the Buffer

Before an application can save an address to the database, it must convert the associated address handle into an *address number*. This address number is then used as a database key for the address. As soon as the application is assigned a number for an address, the handle becomes useless. From then on, the address is only accessed through its address number. However, you use the same function modules (there are parameters in the function module interface for both address handles and address numbers).

If the application uses the address number to access an address, this does not mean that operations are executed directly in the database. All function modules of function group `SZA0` operate on the local memory. To transfer the changes to the database, CAM provides Function Modules in Function Group SZA0 [Page 23].

Address and Application Object

The application links the address to the application object through its address number. To do this, the application adds a field (address type 1) to its application table by saving the address number after it has been assigned. For address types 2 and 3, the application needs two fields since both the key for the person and the key for the address are stored in the table record.

The section on Maintaining Addresses [Page 30] provides more details on the steps an application must perform to maintain addresses using CAM.

Tables of CAM

All addresses are stored in tables of central address management. To be able to access addresses later on, the application stores only the key of an address in its application table.

Some applications that used to store address data in their own tables in previous releases still do so and keep the address data redundantly in their own tables.

The application program never accesses the tables of CAM directly, but uses function modules to access addresses. Using the table entries, however, you can verify if the address data has been updated correctly. The address data is stored in three tables:

Address Data Tables of Central Address Management

Table	**Type**	**Address data**		
		of address type 1	***of address type 2***	***of address type 3***
ADRC	Address table	Organization name and postal address	Postal address	Company address for person
ADRP	Person table	*none*	Personal data (such as last name and first name)	Personal data (such as last name and first name)
ADCP	Assignment of persons to addresses	*none*	*none*	Workplace data

An address number in table **ADRC** or a person number in table **ADRP** is unique. However, both a person number and an address number can occur more than once in table **ADCP**. A person number occurs more than once if a person has multiple addresses, and an address number occurs more than once if a company address is referenced by multiple persons.

The numbers are only unique in one client of a system. For more information, see the section *Uniqueness of Address Numbers* in Transports Prior to Release 4.6 [Page 78].

Assignment of Persons to Addresses

When you create personal or workplace addresses, CAM stores the personal data and the address data separately. Using an entry in table **ADCP**, CAM assigns the address data to the person:

- A record of the table contains the person number and the associated address number.
- Field **COMP_PERS** has either value 'C' (company address, address type 3) or 'P' (personal address, address type 2)

The Where-Used List

Definition

A where-used list of an address indicates which database record of an application table references the address. If not specified otherwise, the statements given below also apply to the where-used list for persons (for address types 2 and 3).

Since the where-used list *references* an application object, it is also commonly called a *reference*.

Depending on whether the application object is the owner of an address or not, the associated where-used list is called *owner reference* or *usage reference* (see Designated Application Objects [Page 21]).

Usage

The application must specify a where-used list when it assigns a number to an address. Based on the application table name, the table field for the reference to the address, and the database key of the corresponding record in the database, the address can be exactly assigned to the application object. This is important for:

- Address data *consistency checks*: In particular, these checks can determine if references to an address exist. If this is the case, the address cannot be deleted.
- Address data *distribution*: The address numbers in the application tables point to addresses in CAM tables. For master data, the where-used list is a means of determining the application object that points to the address. This is important for distribution since address numbers are only unique within one system and therefore must be frequently reassigned in the target system. (See also: Transport of Customizing Addresses [Page 75], Inbound Processing [Page 108] and Distributing Addresses Using ALE [Page 94]).
- *Application-specific enhancements*: CAM offers function modules, for example, that are used by SAPphone to determine the relevant business partner (that is, the application object) based on a telephone number. Without the where-used list such a function could not be implemented.

Prerequisites

The application must register CAM usage in table **`TSADRV`** which contains data about the usage of the address. Based on this data, CAM checks for each reference passed if the application has filled the structure correctly. This prevents program errors on the application side.

Due to the **`TSADRV`** entry, developers can also store information on an associated BOR object (see Preparing to Work With CAM [Page 28]).

The Address Group

Definition

The address group is an attribute for an address or a person. By means of this group, an address is assigned to a person or an application.

Usage

Since the address data is stored centrally, the CAM tables contain addresses from different applications. If an application searched the address data without using a filter, the search result would also include addresses from other applications. This is why CAM uses *address groups* to divide the address data into logical groups. An address can be assigned to more than one group. This filter prevents the search process from returning addresses that do not belong to the application.

Integration

Application developers who want to make use of CAM should clarify at an early stage if their application can be assigned to an existing group or if a new group is required. Address groups are stored in table **`TSAD7`**, and person groups are stored in table **`TSAD8`**.

Application developers must make the necessary group assignments for standard applications in cooperation with the CAM developers at SAP. Customers can define groups in the customer namespace for their own applications.

Designated Application Objects

The Owner of an Address (Parameter `OWNER`)

The application can define one or more application objects as the *owner* of an address. Multiple owners are required if different views for a business object exist.

In the *site master*, for example, the customer, the vendor and the plant are owners of one and the same address (this means that the application objects point to the same address number). In the *user master*, the logon user and the office user point to the same address.

CAM registers the owner using an indicator in the where-used list. In the default setting, this indicator is set automatically when the address number or person number is assigned (that is, when the address is created). The associated where-used list [Page 18] is then called the *owner reference*. Only the owner of an address is authorized to delete the address. Normally, all address maintenance tasks are also performed exclusively by the owner.

In many cases, master data objects are the owners of an address.

Normally, each address has an owner since it cannot be deleted if it does not have one. Exceptions are described in the next section.

Adresses Without Owners

There are application scenarios where all application objects that reference an address have the same authorizations (this is normally true for movement data). In such cases it does not make sense to have a designated owner. To ensure this, the application must set parameter **`OWNER`** to **`SPACE`** when the address number is assigned. The associated where-used list [Page 18] is then called the *usage reference*.

In SD, multiple documents or items have a link to the same address but none of these documents or items is designated as the owner even not the document or item that accidentally assigned the address number first.

In SAPoffice, socalled "direct addresses" (person group **`SODI`**) are created during send and receive.

CAM does not allow the last where-used list of an address to be deleted. The last application object that holds a reference to the address must delete the entire address including this reference.

See also Deleting Where-Used Lists and Addresses [Page 42].

Addresses Without Application Object

In some application-specific processes, the address exists **before** the associated application object exists since the address could or should not be assigned at the time it was entered. Other application objects can only have a link to such addresses (where-used list).

In SAPoffice, the following addresses are entered in a separate transaction and without an application object:

External communication partners using transaction `SOAD` (company addresses)

External communication partners using transaction `SOCP` (workplace addresses)

Transaction `MEAN` is used to enter permanent delivery addresses.

Although the addresses are initially stand-alone, a where-used list must nevertheless be filled when you create them. This is done when you create addresses in the full-screen transaction [Page 49] of CAM. CAM fills the where-used list as follows:

Type	Field	Value
Where-used list for addresses (`ADRV`)	`APPL_TABLE`	`ADRC`
	`APPL_FIELD`	`ADDRNUMBER`
Where-used list for persons (`ADRVP`)	`APPL_TABLE`	`ADRP`
	`APPL_FIELD`	`PERSNUMBER`

Working With CAM

This section is exclusively intended for developers. It describes the basic steps necessary to manage addresses using CAM:

- Planning
- Maintaining addresses
- Maintaining where-used lists
- Using standard dialogs

All CAM-related functions and data are encapsulated in development class **SZAD**. The next sections provide an overview of the development class functions that you use to perform the above-mentioned tasks.

Demo report **EXADR1** in development class **SZAD** illustrates CAM usage based on a small-scale application scenario.

Function Groups

In order to manage addresses using CAM, you basically need function modules of four function groups:

- Function group **SZA0** encapsulates functions for managing addresses of all address types without a dialog (CAM uses these function modules for implementing the standard dialogs). In this context, the function modules access the local memory [Page 14] (internal tables of this function group).
- Function groups **SZA1**, **SZA5** and **SZA7** provide standard dialogs (subscreen, dialog box) for address types 1, 3 or 2. See the section on Using Standard Dialogs of CAM [Page 49].

Overview of Important Function Modules

The prefix of a function module of development class **SZAD** refers to either one of the following:

- The *address* object in general (prefix **ADDR**) or the *person* object (prefix **ADDR_PERSON**). In the first case, the function refers to all or only some address types (which is indicated in the text).
- To an *address type*:
  - Prefix for address type 1 (company addresses): **ADDR**
  - Prefix for address type 2 (personal addresses): **ADDR_PERSONAL**
  - Prefix for address type 3 (workplace addresses): **ADDR_PERS_COMP**

The following table groups function modules by placeholder <*object*> (which means that there is a function module for both objects) or by placeholder <*type*> (which means that there is a function module for each address type). The placeholder used tells you whether prefix **ADDR** refers to the address in general or to address type 1.

Function module overview by function group

Function group	Meaning	
	Function module	***Meaning***
`SZA0`	Central function modules and address maintenance without dialog (all address types [Page 11]).	
	<Object>`_NUMBER_GET`	Assigns a number for an address (of any address type) or a person. You must specify a where-used list when the number is assigned.
	<Object>`_REFERENCE_DELETE`	Deletes a usage for an address (of address type 1 or 2) or a person.
	<Object>`_REFERENCE_INSERT`	Inserts a new usage for an existing address (of address type 1 or 2) or a person.
	<Type>`_COMM_GET`	Reads telecommunication data for an address of type *<type>*.
	<Type>`_COMM_MAINTAIN`	Maintains telecommunication data for an address of type *<type>*.
	<Type>`_DELETE`	Deletes an address of type *<type>* without dialog.
	<Type>`_GET`	Reads an address of type *<type>* without dialog.
	<Type>`_INSERT`	Inserts an address of type *<type>* without dialog.
	<Type>`_UPDATE`	Updates address data without dialog for addresses of type *<type>*.
	`ADDR_MEMORY_CLEAR`	Initializes the local memory of function group `SZA0`.
	`ADDR_MEMORY_SAVE`	Saves all address data from the local memory [Page 14] to the database provided that all handles have been converted to numbers.
	`ADDR_SINGLE_SAVE`	Saves the address data of a **single** address from the local memory to the database.
`SZA1`	Dialogs for company addresses (address type 1)	
	`ADDR_DIALOG`	Dialog box: Dialog-based maintenance of company addresses (create, change, display).

	ADDR_DIALOG_PREPARE	Dialog box, subscreen, full screen: Sets parameters for special cases of address maintenance (field options, titles, and so on).
	ADDR_EXIT_SUBSCREEN	Subscreen: Verifies if address data has been changed (call in module **AT EXIT-COMMAND**).
	ADDR_EXP_SUBSCREEN	Subscreen: Transfers data from application module pool to subscreen (handle, initial values for address fields, and so on).
	ADDR_IMP_SUBSCREEN	Subscreen: Transfers data from subscreen to application module pool (content of address fields, error information, change indicator).
	ADDR_SELECT_FOR_DIALOG	Full screen: Entry point for dialog-based address maintenance as an alternative to parametrized transaction **SADR** (if you need **ADDR_DIALOG_PREPARE** for an application-specific screen configuration).
	ADDR_SUBSCREEN_SET_OKCODE	Subscreen: Passing on an OK code for controlling the subscreen.
SZA5	Dialogs for workplace addresses (address type 3)	
	ADDR_PERS_COMP_DIALOG	Dialog box: Dialog-based maintenance of workplace addresses (create, change, display).
	ADDR_PERS_COMP_DIALOG_PREPARE	Dialog box, subscreen, full screen: Sets parameters for special cases of address maintenance (field options, titles, and so on).
	ADDR_PERS_COMP_EXIT_SUBSCREEN	Subscreen: Verifies if address data has been changed (call in module **AT EXIT-COMMAND**).
	ADDR_PERS_COMP_EXP_SUBSCREEN	Subscreen: Transfers data from application module pool to subscreen (handle, initial values for address fields, and so on).
	ADDR_PERS_COMP_IMP_SUBSCREEN	Subscreen: Transfers data from subscreen to application module pool (content of address fields, error information, change indicator).

	ADDR_PERS_COMP_SELECT_DIALOG	Full screen: Entry point for dialog-based address maintenance as an alternative to parametrized transaction SADR (if you need ADDR_DIALOG_PREPARE for an application-specific screen configuration).
	ADDR_PERS_COMP_SUB_SET_OKCODE	Subscreen: Passing on an OK code for controlling the subscreen.
SZA7	Dialogs for personal addresses (address type 2)	
	ADDR_PERSONAL_DIALOG	Dialog box: Dialog-based maintenance of personal addresses (create, change, display).
	ADDR_PERSONAL_DIALOG_PREPARE	Dialog box, subscreen, full screen: Sets parameters for special cases of address maintenance (field options, titles, and so on).
	ADDR_PERSONAL_EXIT_SUBSCREEN	Subscreen: Verifies if address data has been changed (call in module AT EXIT-COMMAND).
	ADDR_PERSONAL_EXP_SUBSCREEN	Subscreen: Transfers data from application module pool to subscreen (handle, initial values for address fields, and so on).
	ADDR_PERSONAL_IMP_SUBSCREEN	Subscreen: Transfers data from subscreen to application module pool (content of address fields, error information, change indicator).
	ADDR_PERSONAL_SELECT_DIALOG	Full screen: Entry point for dialog-based address maintenance as an alternative to parametrized transaction SADR (if you need ADDR_DIALOG_PREPARE for an application-specific screen configuration).
	ADDR_PERSONAL_SUB_SET_OKCODE	Subscreen: Passing on an OK code for controlling the subscreen.
SZAE	Function modules for complex or summarized address operations	
	*<Type>*_GET_COMPLETE	Reads the complete address data for address of type *<type>*
	*<Type>*_MAINTAIN_COMPLETE	Maintains the complete address data for address of type *<type>*

Preparing to Work With CAM

The Role of the Application Objects

Application objects that use addresses include purchase orders, invoices, sales orders, and so on. In order to use CAM properly, you must answer the following questions:

- Which application objects are linked to an address (which relations exist)?
- Do the application objects that include an address belong to the master data or to the movement data category?
- Do the application objects that reference an address have the same authorizations, or are there objects with some kind of address management function (for example, because you created the address)?

These aspects are closely related to CAM concepts (see Where-Used List [Page 18], Designated Application Objects [Page 21]). The last point is dealt with in more detail below.

Number of References to an Address

If an address is used on a *cross-application* basis, that is, referenced by multiple application tables, then the important question is whether all users of that address have the same authorizations. Do these users reference the address only for a single process (such as an invoice), or do they manage the addresses for an extended period of time (which is the case for customer master data, for example)? Considering these points, it makes sense to grant objects that exist for an extended period more address access authorizations than other objects.

Sometimes addresses do not have a link to an application object. This occurs in applications like SAPoffice that provide address book functionality and is also true for delivery addresses in Purchasing. These are, however, special cases that are currently only supported for address types 1 and 3.

Prerequisites for Using CAM

Determining the Address Group

The address group [Page 20] is an attribute of an address or a person that must be specified by the application as a filter for almost all operations. Before you can use CAM, you must assign your application to an address or person group together with the CAM developers.

The name of an address or person group is made up of two letters for the application ID and two characters that you can choose as required. For example, address group **`ME01`** belongs to materials procurement.

Customers can define their own groups in the customer namespace. The names of these groups are also four characters long and must begin with **`Y`** or **`Z`**.

When you maintain the address or person group, you also maintain the indicator **`MAINT_TYPE`**. If this indicator is set, the addresses can be accessed directly. This means that the addresses that belong to this group may be maintained in the CAM standard transactions [Page 49] without an application object.

Registering the Application in CAM

You must register the application by making an entry in table **TSADRV**. This table stores information on the usage and an associated BOR object (see Maintaining the Where-Used List [Page 37]).

Maintaining Addresses

Preliminary Remark

This section describes what you must do to integrate addresses into your application using CAM. The procedure is described regardless of the address type [Page 11] and is applicable to all dialog techniques [Page 49] except the full-screen mode (see Using CAM Standard Dialogs [Page 49]).

See also the documentation on function group **SZA0**.

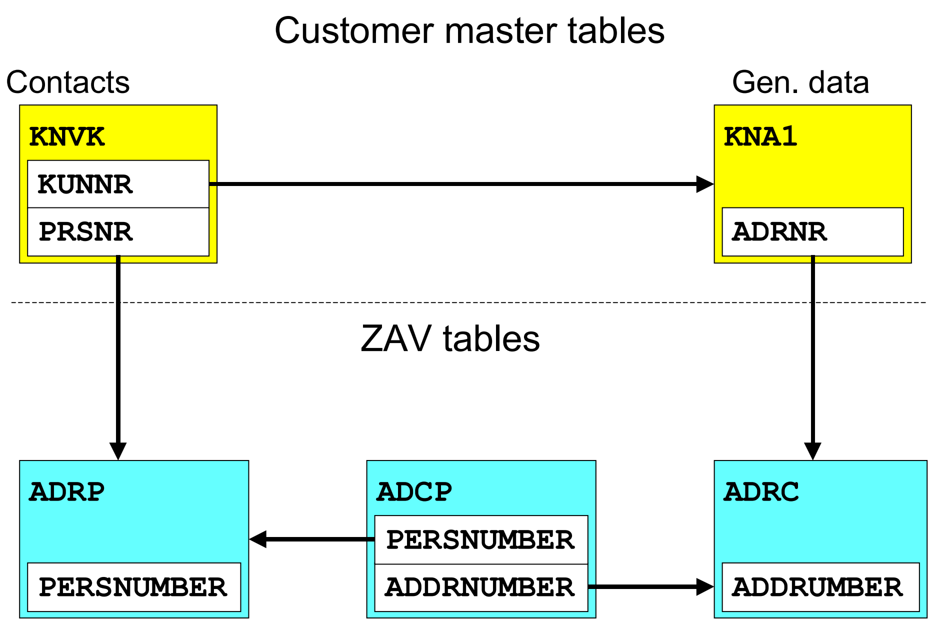
Prerequisites

1. Your application has been assigned to an address or person group and registered in CAM (see Preparing to Work With CAM [Page 28]).
2. You have appended the necessary fields to the structure of your application table whose records should hold links to addresses:
   - For address type 1 (company addresses) you must append a field of type **AD_ADDRNUM** to the structure.
   - For address type 2 (personal addresses) you must append two fields to the structure: a field of type **AD_PERSNUM** (for the person) and a field of type **AD_ADDRNUM** (for the address of that person).
   - If you use address type 3 (workplace addresses) you must append a field of type **AD_PERSNUM** to the structure. You either hold the reference to the company address in the same table, or you use another application object to link to that address (see the example below).

Example

We will use the customer master to illustrate the settings for address type 3.

Table **KNVK** holds data for the *contact* application object, and table **KNA1** stores general customer data. Address group **BP** has been defined for the customer master and maintained in **TSAD7** (address group) and **TSAD8** (person group). Person and address usage has been registered in table **TSADRV**. The customer master stores the person number in table **KNVK** (field **PRSNR**) and the address number for the company address in table **KNA1** (field **ADRNR**). This means that the customer master holds the references for addresses of address type 3 in two different application tables (see figure).


Customer master tables
Contacts
Gen. data
KNVK
KUNNR
PRSNR
KNA1
ADRNR
ZAV tables
ADRP
PERSNUMBER
ADCP
PERSNUMBER
ADDRNUMBER
ADRC
ADDRUMBER

Creating Addresses

Prerequisites

See Maintaining Addresses [Page 30].

Process Flow

1. Define a new handle for the address:

- Address type 1: an address handle
- Address type 2: an address and a person handle
- Address type 3: a person handle (and an address handle if the company address does not yet exist)

  The glossary of the documentation for function group **SZA0** contains a recommendation concerning the convention according to which you should set up your handle.

2. If required, lock your application object and the associated address/person (see also: Notes on Lock Management [Page 51]).
3. If you use a CAM standard dialog [Page 49], set the access mode to **CREATE** when you call the dialog. Otherwise, you use the function module <*type*>**_INSERT** that is appropriate for the address type (see function module overview in Working With CAM [Page 23]). In both cases, you pass the handle that you defined in the first step to CAM.

  If you want to enter more than one address (before saving the data to the database), start again with step one.

  There are function modules for address type 2 and 3 that you can use to create an entirely new address or maintain additional address data for a person.

Having entered and checked the address and application data, you convert the address/person handle into an address/person number:

4. Fill a structure of type **ADDR_REF** for the where-used list of the address and assign an address number using function module **ADDR_NUMBER_GET**.
5. If you use an address of type 2 or 3, also fill a structure of type **PERS_REF** for the where-used list of the person. You assign the person number using function module **ADDR_PERSON_NUMBER_GET**.

You now update the address data together with the application data.

6. Fill the fields that contain the address/person references (in your application table) with the numbers assigned and save your application data.

7. If on account of your dialog handling, you can enter only one address, call function module **`ADDR_SINGLE_SAVE`**. In all other cases, you must use function module **`ADDR_MEMORY_SAVE`**. Then use **`COMMIT WORK`** to update the address data together with the application data.

    You can use function modules **`ADDR_MEMORY_PUSH`**, **`ADDR_MEMORY_POP`** and **`ADDR_MEMORY_RESTORE`** to implement an undo function. You use **`ADDR_MEMORY_CLEAR`** to initialize the local memory.

8. If you have set any locks in the second step, do not forget to remove them.

Result

Depending on the address type, you find the address records saved and the associated where-used lists [Page 37] in the corresponding tables [Page 16].

Changing Addresses

Prerequisites

To access an address, you can identify it using either:

- The handle for the address that you have passed to CAM, or
- The address number that you assigned before you saved the address.

Consequently, pass either the handle or the number to the function module (the interface supports both methods).

In addition, the prerequisites described in Maintaining Addresses [Page 30] must be fulfilled.

Process Flow

1. If required, lock your application objects and the associated address/person (see also: Notes on Lock Management [Page 51]).
2. If you use a CAM standard dialog [Page 49], set the access mode to **CHANGE** when you call the dialog. Otherwise, you use the function module <*type*>**_GET** that is appropriate for the address type (see function module overview in Working With CAM [Page 23]). In both cases, you pass the associated address/person number. The dialog displays the current address data, whereas the function module returns a structure with the current data.

   If you want to change more than one address (before saving the data to the database), start again with step one.

Having changed and verified the address and application data, you update the data together with the application data:

3. If you did not use a dialog, you use the function module <*type*>**_UPDATE** that is appropriate for the address type to pass the address data changed.
4. If on account of your dialog handling, you can change only one address, call function module **ADDR_SINGLE_SAVE**. In all other cases, you must use function module **ADDR_MEMORY_SAVE**. Then use **COMMIT WORK** to update the address data together with the application data.

   You can use function modules **ADDR_MEMORY_PUSH**, **ADDR_MEMORY_POP** and **ADDR_MEMORY_RESTORE** to implement an undo function. You use **ADDR_MEMORY_CLEAR** to initialize the local memory.
5. If you have set any locks in the first step, do not forget to remove them.

Result

Depending on the address type, you find the address records changed in the corresponding tables [Page 16].

Displaying Addresses

Prerequisites

To access an address, you can identify it using either:

- The handle for the address that you have passed to CAM, or
- The address number that you assigned before you saved the address.

Consequently, pass either the handle or the number to the function module (the interface supports both methods).

In addition, the prerequisites described in Maintaining Addresses [Page 30] must be fulfilled.

Process Flow

Set the access mode to `DISPLAY` when you call the standard dialog [Page 49]. To identify the address, you use the address/person number from the record of your application table.

Result

The dialog displays the current address.

Maintaining Where-Used Lists

Usage

You must create a where-used list [Page 41] whenever you save a reference to an address or a person in a record of your application table.

See also: The Where-Used List [Page 18]

Prerequisites

The application must register CAM usage in table **TSADRV** before an address or person can be created together with its where-used list. By means of this registration, CAM checks for each reference passed if the application has filled the where-used list correctly (see Structure of the Where-Used List [Page 38]). This prevents program errors on the application side.

In addition, you can use other fields to further specify the where-used list:

- By means of logical table names [Page 40] you introduce an additional reference level so that changes to names of application tables in the DDIC do not affect the where-used list.
- You can specify callback function modules for BAPIs (to determine the address or person number).
- You can fill fields for determining the owner object (BOR object name, offset/length of the BOR key in the application key, or name of the callback function module for determining the object dynamically).

The data elements for the fields of table **TSADRV** provide further information on these settings.

Structure of the Where-Used List

The where-used list [Page 18] for an address (structure **ADDR_REF**) or a person (structure **PERS_REF**) consists of the following fields:

Field name in structure ADDR_REF	Field name in structure PERS_REF	Meaning
APPL_TABLE	**APPL_TABLE**	Name of the application table
APPL_FIELD	**APPL_FIELD**	Name of the field in the application table that stores the address/person number
APPL_KEY	**APPL_KEY**	Key of the application table (including the client)
ADDR_GROUP	**PERS_GROUP**	Address/person group [Page 20]
OWNER	**OWNER**	Indicates whether the application object specified with **APPL_TABLE**, **APPL_FIELD** and **APPL_KEY** is the owner object for this address or person (see: Designated Application Objects [Page 21]).
	PERS_ADDR	Specifies if the person is used for a personal address (**PERS_ADDR = 'X'**) or for a workplace address (**PERS_ADDR = SPACE**).
	ADDRNUMBER	If **PERS_ADDR = SPACE** has been set, you must enter the address number of the company address (address type 1) in this field.

You can find more information on how to fill the where-used list in the documentation on parameter **ADDRESS_REFERENCE** (**PERSON_REFERENCE** for where-used lists for persons) of function module <*object*>**_NUMBER_GET** (see also the function module overview in Working With CAM [Page 23]).

Integration

CAM stores the where-used list for addresses in table **ADRV**, and the where-used list for persons in table **ADRVP**. Depending on whether address data is stored in **ADRC** or **ADRP** (compare CAM Tables [Page 16]), the following where-used lists must be created:

Where-used lists to be saved depending on address type

Address type	Where-used list for person (ADRVP)	Where-used list for address (ADRV)
Company address (1)		X
Personal address (2)	X	X
Workplace address (3)	X	(X)*

(X)*: If you link a new workplace address to an existing company address, you do not need to create a where-used list for this address since the reference to the person makes it implicitly clear where the address is used. If the company address does not yet exist, you must first create it and specify a where-used list.

Logical Table Names in Where-Used Lists

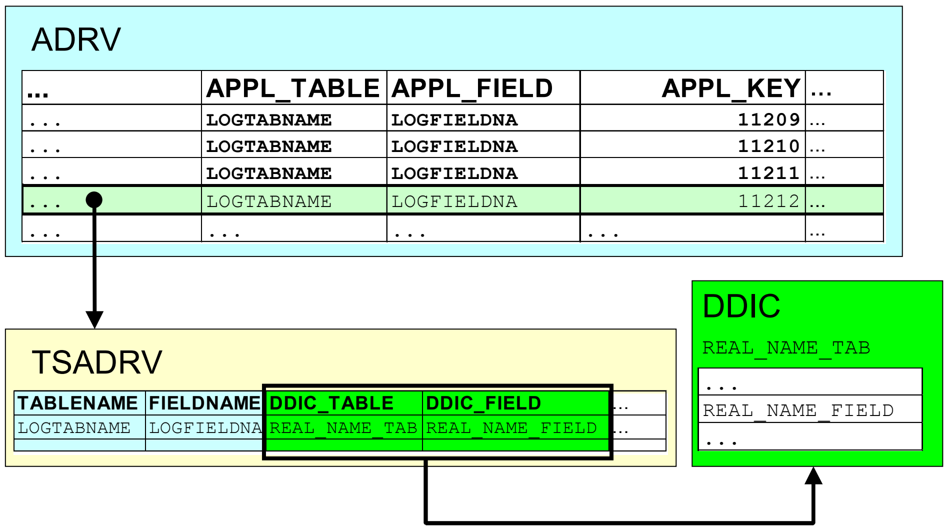
Usage

For Release 4.0, the number of characters allowed for DDIC table and field names was increased to 30. CAM introduced logical table names to prevent where-used lists from being converted.

Using a logical table name provides applications with the advantage that modified DDIC names must be changed only once in **TSADRV**. The entries in the tables for the where-used lists (**ADRV**, **ADRVP**) are not affected by this change.

Assignment of Logical Table to DDIC Table

In the where-used list, the application passes the logical table name. Using the entry in **TSADRV**, CAM can then assign the logical table name to the DDIC table name (the same applies to **ADRVP**):



ADRV

...	APPL_TABLE	APPL_FIELD	APPL_KEY	...
...	LOGTABNAME	LOGFIELDNA	11209	...
...	LOGTABNAME	LOGFIELDNA	11210	...
...	LOGTABNAME	LOGFIELDNA	11211	...
...	LOGTABNAME	LOGFIELDNA	11212	...
...	...	...	...	...

TSADRV

TABLENAME	FIELDNAME	DDIC_TABLE	DDIC_FIELD	...
LOGTABNAME	LOGFIELDNA	REAL_NAME_TAB	REAL_NAME_FIELD	...

DDIC

REAL_NAME_TAB

...
REAL_NAME_FIELD
...

Activities

Normally, new CAM users use the same logical name and DDIC name except if the DDIC table name or field name has more than 10 characters. If you want to change the DDIC table (or field) name later on, you must also change the field **DDIC_TABLE** (**DDIC_FIELD**) of your **TSADRV** entry.

Creating Where-Used Lists

You create a new where-used list for an address or person in the following cases:

- You assign a new address or person number.
- You save a new reference to an existing address or person, for example, if you add a workplace address for an existing person or enter a document that refers to an address.

The Where-Used List During Address/Person Number Assignment

You pass the structure of the where-used list using a parameter of function module <*object*>_**NUMBER_GET** (see Working With CAM [Page 23]). In the default setting, CAM automatically sets the owner indicator [Page 21] (parameter **OWNER**) when the number is assigned even if the structure passed specifies something different. If the address should not have an owner, you must set function module parameter **OWNER** to **SPACE**.

Whether an address has an owner or not is important when it comes to deleting the where-used lists and the address or person itself (see Deleting Where-Used Lists and Addresses [Page 42]).

Creating Additional Where-Used Lists

In order to create additional where-used lists, you use function module <*object*>_**REFERENCE_INSERT**. You must delete these where-used lists first before you can delete an address.

Deleting Where-Used Lists and Addresses

A record of an application table points to an address using the address number and/or person number, if applicable. When the record is deleted, the associated where-used list must be also deleted. From the point of view of the address or person, we must distinguish between two scenarios:

- The address has one ore more owners [Page 21].
- The address does not have an owner.

Depending on the scenario, you must use a different procedure for deleting addresses and where-used lists. The sections Deleting Addresses With one or More Owners [Page 45] and Deleting Addresses Without Owners [Page 47] explain how you must proceed.

Function Module Overview

Generally, the following function modules exist (compare Working With CAM [Page 23]):

Function Module	**Meaning**
`ADDR_REFERENCE_DELETE`	Deletes a where-used list in **`ADRV`** (address types 1/2).
`ADDR_PERSON_REFERENCE_DELETE`	Deletes a where-used list in **`ADRVP`** (address types 2/3).
***<Type>*`_DELETE`**	Deletes an address of type *<type>* and/or a where-used list.
***<Type>*`_VERSION_DELETE`**	Deletes an international version for the address of type *<type>*.

For information on the precise behavior of these function modules, see the sections Function Modules for Deleting References [Page 43] and Function Modules for Deleting Addresses [Page 44].

Function Modules for Deleting References

CAM provides two function modules for deleting *references*:

- Function module **`ADDR_REFERENCE_DELETE`** deletes entries from **`ADRV`**, that is, where-used lists for address types 1 and 2.
- Function module **`ADDR_PERSON_REFERENCE_DELETE`** deletes entries from **`ADRVP`**, that is, where-used lists for address types 2 and 3.

The function modules delete owner references only if other owner references exist for the same address or person. A usage reference is deleted if other references exist for the address or person (regardless of whether these are owner or usage references). If the prerequisites for deleting a reference are not fullfilled, CAM returns an error message.

CAM checks thus ensure that it is allowed to delete an address because either:

- Another owner reference exists (for addresses with one or more owners [Page 45]) or
- Another usage reference exists (for addresses without owners [Page 47])

Function Modules for Deleting Addresses

You delete an address using the function module <*type*>_**DELETE** that is appropriate for the address type. To delete an address, you pass the associated where-used lists to the function module.

If you use function module <*type*>_**DELETE** although additional references for the address exist, the function module attempts to delete the reference(s) passed. It behaves like function modules <*object*>_**REFERENCE_DELETE** for deleting references [Page 43].

⚠

Unlike function modules <*object*>__**REFERENCE_DELETE**, function modules <*type*>_**DELETE** do not issue an error message if the last reference is deleted together with the address.

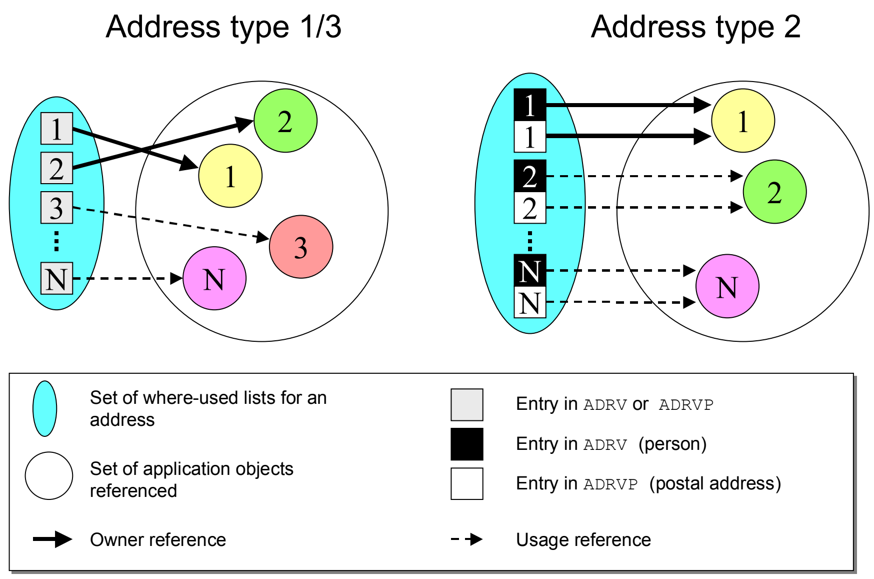
The individual function modules behave as follows (compare CAM Tables [Page 16]):

Address data to be deleted	**ADDR_DELETE**	**ADDR_PERSONAL_DELETE**	**ADDR_PERS_COMP_DELETE**
Record in address table (**ADRC**)	Deleted if only one reference is left for the address and no more persons are assigned (through address type 3).	Deleted if only one reference for the address record is left.	(Nothing to be deleted here, since the workplace address points to an existing company address.)
Record in assignment table (**ADCP**)		Similar to **ADRC** record.	Deleted if only one reference to the person (type 3) is left.
Record in person table (**ADRP**)		Deleted if only one reference to the person and only one assignment in **ADCP** (types 2/3) is left.	Deleted if only one reference to the person and only one assignment in **ADCP** (types 2/3) is left.

Deleting Addresses With one or More Owners

Background

When assigning the address/person number, CAM automatically sets the parameter **OWNER**, making the application object that creates the address the owner of the address. The associated where-used list therefore is an *owner reference*. In addition to this reference, other applications can add owner or usage references for the same address:



The figure shows a sample scenario of where-used lists for an address of type 1, 2 and 3 (address types 1 and 3 are grouped together).

For addresses of type 1 (type 3), you must create a where-used list in **ADRV** (**ADRVP**). In the figure, there are two owner references and other usage references.

For a personal address (address type 2), you must create an entry in both **ADRV** and **ADRVP**. In the figure, one owner reference and other usage references have been created for the address.

Prerequisites

Before the address can be deleted, all additional references must be deleted (see Function Modules for Deleting References [Page 43]). The last reference left must be an owner reference.

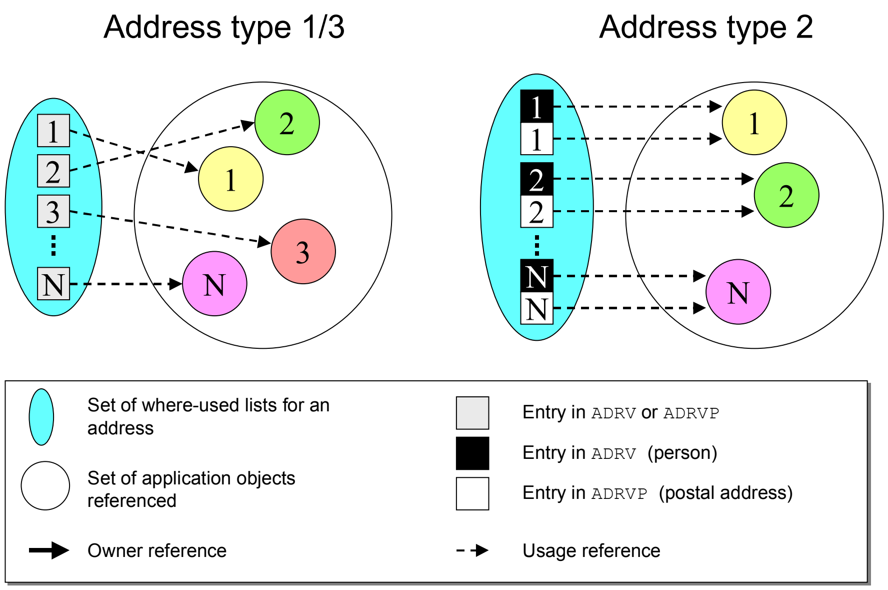
Process Flow

1. Applications that hold usage references to the address use function module <*object*>**_REFERENCE_DELETE** to delete the record in your application table that points to the address.
2. Applications that want to delete the address use function module <*type*>**_DELETE**.
3. Save the changes to the database using function module **ADDR_MEMORY_SAVE**.

Deleting Addresses Without Owners

Background

The application has set the parameter **OWNER** to **SPACE** when calling function module *<object>*_**NUMBER_GET**. All other application objects that point to the same address created where-used lists also without setting the owner indicator (see figure). In this case, all application objects have the same authorizations.



The figure shows a sample scenario of where-used lists for an address of type 1, 2 and 3 (address types 1 and 3 are grouped together).

For addresses of type 1 (type 3), you must create a where-used list in **ADRV** (**ADRVP**). For a personal address (address type 2) you must create an entry both in **ADRV** and **ADRVP**. All application objects are identified through usage references.

Process Flow

Before the address can be deleted, all additional references must be deleted. Since usage references are the only references that exist, a usage reference is left. If an application wants to delete your usage reference but not the address it normally uses the function modules *<object>*_**REFERENCE_DELETE** (see Function Modules for Deleting References [Page 43]).

If you as the user of this address want to delete the address, you must proceed as follows:

1. Attempt to delete the address using function module *<type>*_**DELETE** (pass your usage reference). The function module behaves as follows:
   - If other references to the address exist, the module only deletes the reference.
   - If the reference is the last reference left, the function module deletes both the reference and the address.

2. Save your changes using function module **`ADDR_MEMORY_SAVE`**.

Using CAM Standard Dialogs

Usage

You do not need to create your own maintenance screens for maintaining addresses. If possible, use the CAM standard dialogs that ensure consistency for users that work with addresses in different applications.

Integration

CAM provides three dialog techniques for managing addresses:

- Subscreens
- Dialog boxes
- Full screens

You can use all three techniques to create, change and display addresses. In addition, you can customize all standard dialogs for each application (you can hide functions, field options, and so on). To do this, you call the function module <*type*>**`_DIALOG_PREPARE`** that is appropriate for the address type (see also Working With CAM [Page 23]) before you start the dialog.

Features

This section gives a short introduction to the process of using and programming the standard dialogs. For more information, see the documentation on function group **`SZA1`** (address type 1), **`SZA5`** (address type 3) and **`SZA7`** (address type 2).

The Subscreen

You basically use a subscreen if the address is usually maintained together with the higher-level application object. The interaction of the CAM subscreen with the application that uses the subscreen is controlled by several function modules.

The Dialog Box

You basically use a dialog box if the address is an optional attribute of the application object. You use the function module <type>**`_DIALOG`** that is appropriate for the address type (see Working With CAM [Page 23]).

The Full Screen

The full screen is used for addresses that the user enters without reference to the application object. In a full-screen dialog, you can only maintain addresses for which the field **`MAINT_TYPE`** of the address/person group is set (direct access allowed). This mode includes an address search.

There are two ways to use the full-screen dialog in the program:

- Parametrizing transaction **`SADR`** (address type 1), **`SADQ`** (address type 2) or **`SADP`** (address type 3). Using the parameters, you pass the address/person group to your application.
- Calling function module **`ADDR_SELECT_FOR_DIALOG`** (address type 1), **`ADDR_PERSONAL_SELECT_DIALOG`** (address type 2) or **`ADDR_PERS_COMP_SELECT_DIALOG`** (address type 3).

Only in the last case can the full-screen dialog be customized for a specific application.

Notes on Lock Management

If addresses are changed only through the maintenance dialogs in the full-screen mode, CAM controls the lock management for the addresses. If the application inserts the address fields using the standard subscreen or the standard dialog box, or if the application changes addresses without a maintenance dialog, you must consider the following points:

- If an address of the application is always assigned to one application object only, you can implicitly lock the address by locking this object.
- In all other cases, CAM provides function modules with the postfix **`_ENQEUE`** and **`_DEQUEUE`** for persons and addresses that are also used internally by CAM.

Print Formatting for Addresses

In the course of business processes, addresses are used in letters, forms, and so on. Therefore, one basic requirement put on central address management is to provide routines during printing to ensure that addresses are correctly formatted according to postal conventions.

To this end, CAM provides the function module **`ADDRESS_INTO_PRINTFORM`** which is used by a large number of applications. This function module also controls the print formatting of addresses in CAM standard dialogs.

Addresses are formatted based on valid national and international guidelines:

- ISO 11180
- Contracts of the Universal Postal Union
- International address patterns of the Universal Postal Union
- Regulations of individual countries
- 1999 Guide to Worldwide Postal-Code & Address Formats (Marian Nelson, ISSN 1072-3862).

CAM adopts any changes made to these guidelines at the earliest time possible. If new regulations require the formatting routines to be modified, create a notification in SAPNET and add the official regulation as a reference.

Currently, about 20 country-specific formatting routines are available, for example, for the US, Canada, Germany, Japan, Great Britain, Australia, and so on.

For information on print formatting, also see the composite note 35931 in SAPNet.

Modifications to Standard Formatting

Both applications and customers can adjust standard print formatting to meet individual requirements:

- Applications use parameters of the function module **`ADDRESS_INTO_PRINTFORM`** to override the default settings.
- Customers can use parameters in Customizing and a customer exit [Page 69] to create their own formatting routines.

The Print Preview

The standard dialogs of CAM all provide a `Print preview` pushbutton with which users can display the formatted address in a dialog box. This allows users to check the formatting of the address already when they maintain the data, makes it easier to intuitively understand the meaning of the different fields by means of the print screen and thus supports the field description of the keywords specified.

In addition, users can vary the following parameters:

- Sender country (default setting: the country specified in user parameter 'LND‘ or the country in the user address; if none of these two values has been maintained, the default is 'US‘)
- Printing of street or P.O. box addresses (default setting: P.O. box address – if available; otherwise street address)
- Number of lines available for printing (default setting: 10 lines)

If these parameters are maintained, the address is reformatted and displayed accordingly.

Addresses are displayed in the print preview based on the address formatting of the function module **`ADDRESS_INTO_PRINTFORM`**. Applications that do not use the CAM standard dialog call the function module **`ADDRESS_SHOW_PRINTFORM`** to display the print preview.

Users print the address displayed using the print function in the dialog box.

The formatting of an address in a form or a print program may be different from that in the print preview if the address number is not used for the call and if not all fields maintained are passed in the print program.

Default Settings

CAM determines the formatting routine of an address based on the recipient country. To enable the system to do this, you assign a formatting routine to each country using the *address layout key* field in Customizing [Page 69]. The field accepts a three-digit numeric key. For many countries, this key has already been predefined in the standard SAP system.

This key is called *formatting key* in the following.

There are two number ranges:

- 001 to 899: Reserved for country routines delivered by SAP
- 900 to 999: Reserved for customer-defined formatting routines. If such a value has been assigned to a country in Customizing and an address in which that country is used is formatted, a customer exit [Page 69] is called instead of the function module **ADDRESS_INTO_PRINTFORM**.

For countries for which no formatting key has been maintained, the system uses a standard format that corresponds to formatting routine 010.

In the country settings, you can also maintain the *vehicle country key*. This key is then used as a postal code prefix for some European countries (see Country Formatting [Page 64]).

General Formatting Rules

Normally, address formatting is based on the recipient country. The recipient country is the country that has been entered for the address. CAM calls the formatting routine for that country which formats the address according to the guidelines in place. When an address is formatted, the routine reads the fields of the address and assigns them to *output lines*. These lines are identified by CAM by means of *line IDs*.

Except for some special cases, the formatting routine arranges the output lines using a basic pattern [Page 56]. The application or the user determines the number of lines to be output. If this number is smaller than the number of output lines for an address, the routine must use fewer output lines of the address. The line IDs determine a default priority for suppressing output lines, but the application can redefine this setting (see also the documentation on function module **ADDRESS_INTO_PRINTFORM**).

Literals

Language-specific parts, such as the title and the P.O. box are output in the language of the recipient.

The language of the recipient is the correspondence language maintained for the address. If no correspondence language has been maintained or if no such language is passed to the print function module, then the language of the recipient country (as defined in the country table) is used.

Only if the system is unable to determine the recipient language, the logon language is used.

Basic Pattern

For all formatting routines, except those for Great Britain (006), Japan (013) and South Korea (017), the address is basically formatted as follows (blank lines are dropped, except for the required blank line):

- Title, if required
- Name block [Page 57]
- Street or P.O. box [Page 58]
- Blank line, if required
- City with postal code [Page 63]
- Country [Page 64], if required

For addresses where the recipient country and the sender country are different, the city and the country lines are printed in uppercase.

In some countries, no blank line is required.

Name Formatting

Depending on the type of address, the name consists of the following fields:

Address type	**Output lines**
Company address (1)	NAME1
	NAME2
	NAME3
	NAME4
Personal address (2)	Title + name of person
Workplace address (3)	NAME1
	NAME2
	NAME3
	NAME4
	Department
	Title + name of person
	Function

Street and P.O. Box Formatting

An address consists of several postal attributes (address fields). These attributes can be assigned to either a street address or a P.O. box address. The address itself contains all address fields. However, when the address is printed, you must decide if you want to use the attributes of the street address or of the P.O. box address.

This difference is indicated in address maintenance by means of two different group boxes for the street and the P.O. box address.

Input Fields

The following fields belong to the street address and P.O. box address:

Street address	**P.O. box address**
Street 2	• P.O. box
Street 3	
Street, house number, supplement	
Street 4	• Indicator `P.O. box without number` (if the P.O. box does not have a number and only the words 'P.O. box' are to be printed)
Street 5	
District	
Other city	
Postal code, city	Postal code of the P.O. box/company postal code, other city of P.O. box
Region	Other region
Country	Other country

You must only specify the fields for another city, region or country if these are not identical to the corresponding fields of the street address. If they are identical to these fields or if the fields of the street address are not filled, then you must enter these three fields directly in the street address.

There are also so-called major customer addresses. These addresses have a postal code of their own which must be entered into the field for the company postal code. If you enter such a postal code, the fields for the P.O. box and the P.O. box postal code need not be filled.

Selecting the Address Type

In the print preview [Page 53] of the address, users determine if they want to output the fields of the street or of the P.O. box address. The application controls this using the parameter **`STREET_HAS_PRIORITY`** of the function module **`ADDRESS_INTO_PRINTFORM`**. If this parameter is set, then the fields of the street address [Page 60] are used for formatting. Otherwise, the P.O.

box address is used (in this case, no street lines are required). The system checks in advance if the relevant fields are filled.

Street Line Output

The street lines are printed one after another (provided that a sufficient number of lines is available) according to the following pattern:

- Content of field Street 2
- Content of field Street3
- Street line
- Content of field Street 4
- Content of field Street 5

The fields Street 2 to Street 5 enable you to use other required parts of address lines (for example, building, block, sector, square, floor, apartment, and so on) in various countries flexibly, without having to specify their meaning unchangeably.The field print output positions are as specified above.

In contrast, the meaning of the field 'Street' is clearly defined and unambiguous – as it is interpreted as a unit of structure in most western countries. In addition, the Street field is used for searches (that are not case-sensitive) and can be checked against postal codes (city and street directory), for example.

In some countries, the district may be printed above the city line or the street lines.

Street Line Formatting

The street line comprises the fields Street, House number and House number supplement.

In the Anglo-Saxon countries the fields are in the order House number, Street, House number supplement, while in most European countries the order is Street, House number, House number supplement.

When you maintain an address, you can make an entry into the fields Building code, Room and Floor, for example, to allow searches. However, these fields are not printed. If you want to print the building, room and floor information, you must enter it into the fields Street 2 to Street 5 in exactly the output format you require.

Examples

Output	Address fields
SAP America Inc. 701 Lee Road, Suite 600 Wayne PA 19087 USA	• House number: 701 • Street: Lee Road • House number supplement: Suite 600
SAP Iberoamerica S.A. Torre Mapfre Carrer de la Marina 16-18, 11 B/C E-08005 Barcelona (SPAIN)	• Street 2: Torre Mapfre • House number: 16-18 • Street: Carrer de la Marina • House number supplement: 11 B/C
SAP Asia Pte Ltd. 750A Chai Chee Road 7th Floor Chai Chee Industrial Park Singapore 469001	• House number: 750A • Street: Chai Chee Road • Street 4: 7th Floor Chai Chee Industrial Park
Ministerio de Comunicaciones Es Carrera 8a Entre Calles 12A y 13 Edificio Murillo Toro - Piso 5° SANTAFE DE BOGOTA, D.E.1 (COLUMBIA)	• Street 2/3: <empty> • House number: <empty> • Street: Es Carrera 8a (8a is not the house number in this case, but the number of the carreras = alleys) • House number supplement: <empty> • Street 4: Entre Calles 12A y 13 • Street 5: Edificio Murillo Toro - Piso 5°
SAP Italia S.p.A. Centro Direzionale Colleoni Viale Colleoni 17 Palazzo Orione 3 I-20041 Agrate Brianza/Milano (ITALY)	• Street 2: Centro Direzionale Colleoni • Street 3: <leer> • House number: 17 • Street: Viale Colleoni • House number supplement: <empty> • Street 4: Palazzo Orione 3 • Street 5: <empty>

Examples

SAP Hong Kong Suite 1111-1114, 11/F Cityplaza 4 12 Taikoo Wan Road Taikoo Shing HONGKONG	• Street 2: Suite 1111-1114, 11/F • Street 3: Cityplaza 4 • House number: 12 • Street: Taikoo Wan Road • House number supplement: <emtpy> • Street 4: <empty> • Street 5: <empty>

City Formatting

In the city line, the city and the district are printed as a composition joined by a hyphen (exceptions: 004 USA, 006 Great Britain/Ireland, 013 Japan, 015 Germany, 017 South Korea, 019 Denmark) provided that the total length does not exceed 35 characters.

If another city has been specified for the P.O. box, this city is used for the P.O. box address.

The order and structure of the postal code, city, district and region in the city line differs considerably from country to country.

Country Formatting

The format of the address depends on whether the sender and the recipient country are identical or not. If the countries are not identical, the country is always indicated, either by its full name or by its vehicle country key [Page 54].

If the full country name is used, the formatting routine uses the language of the sender country. If no language is specified for the sender country or if the sender country is not indicated, the logon language is used instead. Applications can override these default settings using parameters of the function module **`ADDR_INTO_PRINTFORM`**.

Country-Specific Country Formatting

The formatting routines for the formatting keys `001` (European standard formatting), `002` (Italy), `011` (Switzerland) and `014` (Austria) use the vehicle country key of the relevant country. If no vehicle country key has been maintained in Customizing, the country key of table `T005` is used instead.

If, for the sender country, the indicator for printing the country name in foreign addresses is set in table `T005`, the system does not use the country key, but generally prints the country name in the last address line.

Usage of the Region Field

In formatting routines `002` (Italy), `004` (USA), `005` (Canada), `006` (Great Britain), `007` (Brazil), and `009` (Australia), the address field **`REGION`** (region, federal state, province, county) is considered when the address is formatted. For Italy, the US, Canada, Brazil, and Australia, the system uses the key from table **`T005S`**, for Great Britain, it uses the name from table **`T005U`**.

Usage of the Function Field

In formatting routines `004` (USA), `005` (Canada), and `008` (Singapore), one output line is reserved for the function of the company contact for workplace addresses (line ID [Page 55] 'F'). This line is printed directly after the output line for the name (and the title) of the natural person (line ID 'N').

International Address Versions

You maintain international address versions in Customizing activity *Maintain address version display formats* (located in the IMG under *Basis Components* → *Basis* Services → *Address Management*). Each version is identified by a one-character ID which is part of the database key of the address. Therefore, several versions can exist for an address number. There is a default version for each address that is identified by **SPACE**. All other versions can be maintained by using the pushbutton for international address versions in the relevant dialogs.

Currently, address versions can only be maintained for company addresses (type 1).

So that versions are output during printing, the address number under which the address is saved in the database, must be passed to the print function module. For example, if you use SAPScript for printing, you must specify the address number as a parameter. Internally, the function module **ADDRESS_INTO_PRINTFORM** then controls which version of an address is printed.

In the standard system, printing address versions is only supported for Japan. Besides the standard version, versions Kanji (**K**) or International (**I**) can be printed.

In this context, the following rules apply:

Japan → Japan	Kanji, if maintained; otherwise, default version
Japan → Foreign country	International, if maintained; otherwise, default version
Foreign country → Japan	International, if maintained; otherwise, default version

If you want to use other international address versions in addition to the Japanese versions, you can use the customer exit **SZAD0001** of the function module **ADDRESS_INTO_PRINTFORM**.

Integration Into the Application

Many application programs use the print function module **ADDRESS_INTO_PRINTFORM** of central address management. A large number of print programs call this function module indirectly through SAPScript, for example. In the address window of SAPScript, you can use the control command ADDRESS to set the parameters of **ADDRESS_INTO_PRINTFORM**.

As of Release 4.6C, Smart Forms [Ext.] use this function module as well.

To ensure proper printing, the parameter **ADDRESS_TYPE** must be set correctly by the calling program when the print function module is called. Depending on the address type [Page 11], value 1,2 or 3 must be assigned.

There are two ways to pass the address to be formatted to the print function module:

- Using the address number [Page 14]: This type of call is recommended for all applications that use CAM addresses. For address type 1, parameter **ADDRESS_NUMBER** must be filled. For address types 2 and 3, the person number must additionally be specified using the parameter **PERSON_NUMBER**. It is also possible to format addresses that have not yet been saved. In this case, you must specify the address handle instead of the address number, and the person handle instead of the person number.
- Passing all relevant fields (address attributes) of the address directly to the print function module: This type of call is used by all applications that do not yet use CAM to store their addresses. In order to pass the structure with the address fields for each address type, you use parameter **ADDRESS_1**, **ADDRESS_2** or **ADDRESS_3**.

You can find the documentation for the function module and its parameters in the Function Builder (**SE37**) of the SAP System.

Customer-Specific Settings

Settings in Customizing

In Customizing under *General Settings* → *Set countries* → *Define countries* (transaction **OY01**), you can assign a key for the country-specific formatting routine to each country. You do this in the group box *Address format* in the field *Address layout key* (**T005-ADDRS**). For countries that have a routine contained in the standard system, these fields are predefined accordingly.

It is also possible to define customer-specific formatting routines using the customer exit **SZAD0001**. You must assign a key in the customer namespace from **900** to **999** to these routines.

Another Customizing parameter that affects the printing of addresses is the indicator 'Print country name' (**T005-XADDR**). This indicator is a setting that refers to the sender country of the address (in contrast to the address layout key which is assigned directly to each recipient country). This indicator controls if the country name or an ID is printed in foreign addresses.

Programming Self-Defined Formatting Routines

If you want to program your own formatting routine, you can use customer exit **SZAD0001** in transaction **CMOD**.

You can find the documentation for this exit in transaction **CMOD** or with function module **EXIT_SAPLSADR_001**.

Addresses of Customizing Objects

Before customers can run an R/3 System, several system parameters must be set in Customizing for the applications. The settings of application objects form one business unit. It makes sense to maintain the relevant parameters in a single step. To do this, you use transaction **`SOBJ`** to define Customizing objects. Although the parameters of these objects may be distributed among many database tables, they can easily be maintained using maintenance views. To make usage of these maintenance views consistent, a standardized table maintenance transaction is available. This transaction is called Extended Table Maintenance [Ext.] (**`SM30`**).

To ensure that you can make the settings in a structured way, IMG activities are provided that take you to extended table maintenance. To do this, the system calls a parametrized transaction that starts the table maintenance transaction.

Transaction **`OY01`** (change country global parameters) starts transaction **`SM30`** with maintenance view **`V_T005`**. This Customizing object does not have an address.

Many Customizing objects have an address which is maintained together with the other settings. These addresses are also called Customizing addresses.

Customizing object *Change plants* (transaction **`OX10`**) has an address and uses maintenance view **`V_T001W`**.

Customizing Objects Using Address Functionality

The following Customizing objects in the R/3 core system use address functionality, for example:

Customizing object	Table	Transaction
Company codes	T001	OX02
Company code-dependent address data	T001E	FSAP
Plants/branches	T001W	OX10
Organizational unit: sales offices	TVBUR	OVX1
Routes: transportation connection points	TVKN	VORD
Organizational unit: sales organizations	TVKO	OVX5
Organizational unit: shipping points	TVST	OVXD
Taxes on sales/purchases groups: address	T007F-ADRNR	OBCM
Tax office address	T007F-FAADR	OBCF
Personnel areas	T500P	
Org. unit: transportation planning points	TTDS	
Lockboxes for house banks	T049L	

Address Maintenance in Customizing

Address maintenance in Customizing is a generic service provided by central address management and extended table maintenance [Ext.]. If address maintenance has been integrated for an object, an address icon is displayed in extended table maintenance that can be used to call a CAM maintenance dialog box. During this process, extended table maintenance calls function modules of CAM such as **ADDR_DIALOG** and **ADDR_SINGLE_SAVE**.

Prerequisites

The Customizing table must have been registered in table **TSADRV** (see also: Maintaining Where-Used Lists [Page 37]). Using function module **ADDR_TSADRV_READ** CAM checks in transaction **SM30** if such an entry exists.

Normally, all entries are similar. See **T001-ADRNR**, for example.

When registering the Customizing table, you also specify an address group [Page 20]. For Customizing objects, CAM provides address group **CA01**. Customers should use address group **ZA01** for self-defined Customizing tables. If this address group does not yet exist, it must first be defined before it can be assigned. Also, if customer objects are deleted, you must verify if address group **ZADE** exists.

Customizing addresses of SAP objects belong to address group **CA01** while the Customizing addresses of customer objects belong to address group **ZA01**. When you delete a Customizing object, the address is not deleted for technical reasons. Instead, the address is set to address group **CADE** (or **ZADE** for customer objects).

Integrating Address Maintenance for a Customizing Table

Process Flow

To automatically display address maintenance for a Customizing table, you must carry out the following steps:

1. Add a field for storing the address number to the table or the view. To do this, use domain **AD_ADDRNUM**. Based on the address domain, the system automatically generates an indicator (**OBJH-OBJHASADDR**) in transaction **SE54** which specifies if the table or view contains an address or not.
2. Maintain the **TSADRV** entry as described above in Address Maintenance in Customizing [Page 72].
3. Use transaction **SE54** to generate the table maintenance dialog for transaction **SM30** (see also: BC – Generate Table Maintenance Dialog [Ext.]).

Result

Address maintenance is now an integral part of Customizing object maintenance.

The address dialog box is called when you choose the address icon. If an address has already been maintained for the object, this address appears in either change or display mode on the dialog box.

If no address has been maintained for the object or if no address exists for an address number [Page 14], the system takes you automatically to the screen for creating an address.

When you create an object, the system takes you automatically to the address dialog box at the time you release or save the data to prompt you to maintain the address. You can skip the address dialog box using the *Cancel* function.

You can find examples of address maintenance in Customizing in transactions **OX10** (plants) and **OX02** (company codes). (See also: Customizing Objects Using Address Functionality [Page 71]).

Adjusting the Generic Address Maintenance Dialog

It is possible to adjust address maintenance in the table maintenance dialog generated to meet the specific requirements of an application. To do this, the application modifies the default process flow of table maintenance at fixed pre-defined events. For address maintenance, this is done at event 23 which is processed before the address maintenance dialog is called.

Before transaction **`SM30`** calls function module **`ADDR_DIALOG`** to process the address maintenance screen, global variables can be set in event 23 in order to adjust address maintenance to meet the requirements of the application. These global variables correspond to the interface parameters of function module **`ADDR_DIALOG_PREPARE`** which makes application-specific settings, such as defining field selection control, naming the title line on the address screen, setting the switch for enabling/disabling the communiction types, and so on.

See also: Extended Table Maintenance Events [Ext.]

Transport of Customizing Addresses

The transport link is a central function in Customizing. The Customizing settings of a system or client can be transported to other systems or clients using Customizing requests. Central address management provides methods for transporting the addresses together with their Customizing objects.

Addresses for Customizing objects (such as plant, company code or sales organization) are usually transported by adding the Customizing data changed to a transport request in the maintenance transaction. During this process, the corresponding address data is added automatically to the transport request.

Transports as of Release 4.6

As of Release 4.6, CAM uses a new transport logic that eliminates potential problems which may arise as a result of transporting the addresses together with their address numbers (see Transports Prior to Release 4.6 [Page 78]). It is now no longer the address tables themselves that are transported, but socalled shadow tables [Page 77] containing the address data. These shadow tables are not productive tables but dummy tables that have all the fields of the original tables. Before the transport, method **`BEFORE_EXP_CUST_ADDRESS`** is called. This method writes the addresses to be transported into the shadow tables and creates the associated where-used list.

The transport takes place using the logical transport object **`R3TR TDAT ADDRESS_4.6`**.

Once the transport request has been imported into the target system, the address objects that have been transported are processed by calling method **`AFTER_IMP_CUST_ADDRESS`**. Since the address numbers are normally not identical in the source and the target systems (they have only local validity), address objects are identified by their owners as are addresses distributed through ALE [Page 94]. This requires that 1:1 (1:c) relations exist between objects and addresses. Several Customizing objects cannot point to the same address number.

The references to the application object stored in the key of the shadow tables are used to determine the where-used lists for the addresses. The where-used lists, in turn, are used to determine the address numbers for the transported addresses that are valid in the target system. The addresses are updated with the address data from the shadow tables transported. If an address does not exist, it is created, and the new address number is updated for all references of the primary tables.

If the shadow table does not contain an address entry, then the address was deleted in the source system or the source client. In this case, the address is also deleted in the target system together with the pointers of the application tables.

Shadow Tables (as of Release 4.6)

Shadow table name	Meaning
ADRCS2	Company addresses
ADRCTS2	Texts for addresses
ADRGS2	Assignment of company addresses to additional address groups
ADRVS	Where-used list for company addresses
ADR2S2	Telephone numbers
ADR3S2	Fax numbers
ADR4S2	Teletex numbers
ADR5S2	Telex numbers
ADR6S2	SMTP numbers
ADR7S2	RML addresses
ADR8S2	X.400 numbers
ADR9S2	RFC destinations
ADR10S2	Printers
ADR11S2	SSF
ADR12S2	FTP and URL
ADR13S2	Pagers
ADRCOMCS2	Sequence numbers for communication data
ADRTS2	Texts for communication data

Transports Prior to Release 4.6

Overview

As of Release 1.1., it has been possible to transport Customizing objects together with their addresses. The view maintenance of Customizing objects provides a direct linkt to address maintenance. This functionality is automatically active if the Customizing object has a table field with domain **AD_ADDRNUM** (prior to Release 3.1I: domain **CADRNR** or **ADRNR**).

Up to Release 3.1I, function module **ADDRESS_MAINTAIN** was used to store Customizing addresses in table **SADR**. This table was also used as the address file for other applications. As of Release 4.0, Customizing addresses are stored in the new table **ADRC** and are assigned to address group **CA01**. When you upgrade from Release 3.x to Release 4.0 or higher, Customizing addresses are migrated from table **SADR** to **ADRC** using the XPRA **RSXADR01** or the prestep report **RSXADR05** (see Data Conversion [Page 80]).

For Releases 4.0 and 4.5, the transport takes place using logical transport object **TDAT ADDRESS**. Prior to 4.0, **R3TR TABU SADR/SADR2/SADR3/SADR4/SADR5** is used.

Special Features for Release 4.0

A new feature that was implemented for Release 4.0 is the flag **OBJH-OBJHASADDR** that indicates if a Customizing object has an address (the flag is set) or not: You can use this indicator to reset the property that an object has an address when maintaining the attributes of a Customizing object.

Uniqueness of Address Numbers

Up to and including Release 4.5, the table keys from the source system, that is, the internal address numbers assigned, are used to transport the address data. When assigning the address numbers, CAM retrieves a number belonging to a number range. You define the number range interval for the number range object **ADRNR** in Customizing for central address management (see Maintain address and person number range [Ext.] in the IMG).

Since address numbers are only unique in one client of a system, conflicts may arise during the transport of addresses:

- The address number of an address transported already exists in the target system.
- The address number does not yet exist in the target system but is reassigned at a later time.

To ensure that the address numbers for addresses are unique across multiple systems, you must define the number range intervals so that they do not overlap.

See also note 25182 in SAPNet. This note describes how you must define the number range intervals. The note is valid up to Release 4.5. As of Release 4.6, the new transport procedure is effective which automatically ensures consistency.

Data Conversion

In our context, the term 'address' includes persons (ADRP) for address types 2 and 3 (where the person number is used as an additional key).

Most applications now use central address management (CAM) to benefit from the functions it provides, such as having available all current types of communication or using standardized print formatting based on national and international rules.

New applications or applications with new address functionality added can directly use CAM without requiring conversion. However, applications that used to have their own address management require data conversion since the address information from the application tables must be transferred into CAM tables and structures. If the application tables store address information as well (for example, in the customer and vendor master), this information is matched internally with the corresponding CAM fields.

Generally, applications store a reference to the address key (and a person number, if required) in the application tables to access the address data. During data conversion, existing address numbers are kept. New address numbers are assigned by CAM.

Data is converted during the upgrade by means of *XPRA* reports (XPRA = E**X**ecution of **PR**ogram **A**fter **I**mport) of CAM. These reports are added to a transport request using statement 'R3TR XPRA repname' and are executed automatically during upgrade downtime.

The system logs the actions of XPRA reports. Since XPRAs can be restarted, data that has already been converted does not have to be converted again if the XPRA terminates and is executed again.

Releases Affected

In the standard R/3 System, CAM data conversion is delivered with Releases 4.0 and 4.5.

4.0 data conversion takes place during the following upgrades, for example:

Source release	**Target release**
3.0/3.1	4.0
3.0/3.1	4.5
3.0/3.1	4.6

4.5 data conversion takes place during the following upgrades:

Source release	**Target release**
3.0/3.1	4.5
3.0/3.1	4.6
4.0	4.5
4.0	4.6

Downtime Reduction (Prestep Process)

In order to reduce the XPRA runtime during an upgrade if large amounts of data have to be handled, we recommend that you convert the majority of the data by means of a *prestep* report before the upgrade. Together with the upgrade tools, this report is imported into the system during upgrade preparation (source release) and should be started as a background job before the upgrade. Converting data in prestep mode is supported as of Release 3.0 since the CAM tables were delivered as early as Release 3.0. So that the data records converted in prestep mode can be identified, they are marked with the flag for the data transfer status (**`DUEFL`** = 'X'). The remaining data is converted during upgrade by a CAM XPRA and then also marked with this indicator.

Creation of the Where-Used List

For all addresses converted, a where-used list [Page 18] (for addresses and persons) is created in CAM tables ADRV and ADRVP. This list is used as the basis for referential integrity when addresses are deleted or archived and is also required for identifying the owner object [Page 21] of an address.

The where-used list is also of practical benefit. For example, you can quickly determine to which application the address data belongs. This allows you, for example, to assign business partner data based on the telephone number in case of incoming calls.

Conversion of the Address Groups

During data conversion, each address must be assigned to the appropriate address group [Page 20]. The XPRA report identifies the appropriate address group based on the application table.

For addresses without a higher-level application object [Page 21], the XPRA assigns an address group (for example, group **BC01** for **SADR** addresses to which office users were assigned).

Data Conversion for Release 4.0

Data is converted for Release 4.0 by means of XPRA **`RSXADR01`**. This report performs the data conversion required for an upgrade to Release 4.0 for the migration to central address management and creates a where-used list by means of the address number references.

During data conversion, data is only transferred to the CAM tables. No table contents are deleted.

During upgrade preparation, function module **`ADDR_UPGRADE_PREPARE`** of CAM is called in phase **`JOB_RSCNVAD`**. This function module writes an error to the log if the number of addresses involved exceeds 50,000. In this case, you can convert large amounts of data in prestep mode using report **`RSXADR05`**. This allows you to considerably reduce the runtime of XPRA **`RSXADR01`**. This does not lead to inconsistencies or effect operation.

In this context, please refer to note 82167.

Tables Affected

The tables filled include the new CAM tables for address information (**ADRC**), person-related attributes (address types 2/3 -> **ADRP**, **ADCP**) and the tables for the possible communication types (**ADR2** to **ADR8**).

Table **SADR** is only partially converted in this step since some of the data is only converted for 4.5.

Source table	Target table	Data converted
SADR	ADRC	Company address (address type 1)
SADRP	ADRP, ADCP	Office users/communication partners (address type 3)
SADR2	ADR2	Telephone numbers
SADR3	ADR3	Fax numbers
SADR4	ADR4	Teletex numbers
SADR5	ADR5	Telex numbers
SADR7	ADR7	Remote mail addresses
SADR8	ADR8	X.400 addresses
SADR10	ADR6	Email addresses
USR03	ADRP, ADCP	System users (address type 3)

Applications Affected

Applications Using Address Type 1

- All addresses for Customizing objects, including the following in the standard R/3 System:
  - Company code
  - Plant/branch
  - Sales organization
  - Sales office
  - Shipping point
  - Transportation planning point
  - Transportation connection point
  - Taxes on sales/purchasing groups (address and tax office address)
  - Personnel area
  - Company code - reply slip addresses (balance confirmations)
  - Table **`T5G52`** (tax district and reference details)
  - Table **`J_1BBRANCH`** (CGC branch)
- Materials Management/Purchasing: manual delivery addresses for purchase orders
- Plant Maintenance and Service Management (functional locations, equipments, notification, order)
- Shipment documents (SD/WS)
- **`KANBAN`** (different delivery addresses)
- Delivery addresses for reservations/dependent requirement

Applications Using Address Type 3

- User addresses and SAPoffice user (office user) addresses: The addresses in **`USR03`** and **`SADRP`** are integrated into a common data structure with consistent maintenance interface and functions.
- Addresses for external communication partners (companies and contacts) for SAPoffice

Applications Using Address Types 1 and 3

- All subsequent functions in SAPoffice that access addresses directly, such as external sending and receiving, distribution lists with addresses, substitutes (external),
- SAPconnect (based on SAPoffice)

Applications not Affected

Data conversion for Release 4.0 is not required for applications that:

- Had address functions added for Release 4.0 and directly used CAM
- Used CAM prior to Release 4.0.

For completeness, these applications are also listed below:

Address Functions Added for 4.0

- Additional addresses in Customizing:
  - Company code - EC tax numbers / periodic declarations
  - Lockboxes for house banks
  - Locations for installations
  - Address determination from plant and storage location
- Addresses in purchase requirements (as with purchase orders)
- Material Management/Purchasing: addresses for one-time vendors
- Address in plant maintenance and service orders
- Address in site master (Retail)
- Using CAM in SAPphone, for example, for incoming calls in Service Management

Applications That Used CAM in a Release Prior to 4.0

- Material Management/Purchasing: permanent delivery addresses for purchase orders
- Central business partner entry in several Industry Solutions
- IS-Utilities (business partners, connection objects)
- IS-OIL (Physical Business Locations)
- Foreign Trade (letter of credit processing)

You can find more information in the release notes for Release 4.0.

Data Conversion for Release 4.5

XPRA report **RSXADR11** converts the data during the upgrade to Release 4.5. As with a data conversion for 4.0, the CAM tables **ADRC**, **ADRP**, **ADCP** and the tables for some communication types (**ADR2**, **ADR3**, **ADR4**, **ADR5** and **ADR12**) are filled. During upgrade phase **JOB_RSCNVADR**, the system calls function module **ADDR_UPGRADE_PREPARE_45A**. It additionally calls function **ADDR_UPGRADE_PREPARE** if the source release is a release prior to 4.0.

If you need to convert more than 300,000 data records, we recommend that you convert some data in prestep mode using report **RSXADR21**. Please read note 97032.

Tables Affected

Table **SADR** is only partially converted in this step since some of the data has already been converted during the 4.0 upgrade.

Source table	**Target table(s)**	**Data converted**
Master data		
KNA1	ADRC	Customer master addresses
LFA1	ADRC	Vendor master addresses
KNVK	ADRP, ADCP	Customer contacts (address type 3)
SADR	ADRC	
	ADRV	New: address where-used list
	ADRVP	New: person where-used list
Document addresses (SD)		
SADR	ADRC	
SAIN	ADRV	

Depending on the data constellation, tables **ADR2**, **ADR3**, **ADR4**, **ADR5** and **ADR12** are filled as well.

All data in the source table is kept. No data is deleted. The data transferred to the target tables (in prestep mode) before the XPRA is executed is not deleted either.

Applications Affected

For 4.5, more core applications have been enhanced to use central address management, and a corresponding where-used list has been created.

Applications Using Address Type 1

- Customer master
- Vendor master
- Bank addresses
- Document addresses in SD/WS

Applications Using Address Types 1 and 3

- Customer contacts with their company and private addresses

You can find more information in the release notes for Release 4.5.

Notes on Upgrade Preparation

Note	Subject
165262	Conversion of international address versions (maintain table **`TSAVX`**)
96607	Changes in address management for 4.0A
97032	Prestep for Rel. 4.5: **`RSXADR21`**
82167	Prestep for Rel. 4.0: **`RSXADR05`**
97802	Tablespace sizes for converting addresses for Rel. 4.5

For common problems, you can find notes in SAPNet that are entered as associated notes for the notes listed above.

Distributing Addresses Using ALE

Background

The integration technology Application Link Enabling (ALE) is an important middleware tool in the Business Framework Architecture (BFA). ALE integrates business processes both between SAP Systems, and between SAP Systems and non-SAP systems. Data is exchanged between application systems in a controlled manner, and is kept consistent.

The application systems in an ALE integrated system are loosely coupled. Data is exchanged asynchronously. The system ensures that the data reaches the receiving system even if this is not available at the time the data is sent. ALE uses synchronous connections only for reading data.

The basis for distributed applications in an ALE integrated system is the asynchronous distribution of messages through outbound and inbound processing. Messages are distributed using data containers, called IDocs (**I**ntermediate **Doc**uments).

For an introduction to ALE, see ALE Introduction and Administration [Ext.].

Usage

The following master data objects with addresses can currently be distributed using ALE:

- Customer master
- Vendor master
- SAP business partners
- Bank master

Exceptions: As of Release 4.5, users (BOR object USER) are distributed together with their addresses in a commom IDoc. Company addresses assigned are distributed using the user company (BOR object USRCOMPANY).

Central Address Management and ALE

Addresses of central address management are distributed as independent objects by means of specific IDocs. The BOR object types for the three address types are `BUS4001` (company addresses), `BUS4002` (personal addresses) and `BUS4003` (workplace addresses).

Addresses are distributed separately, but never independently of the address application object. They are not distributed through their key (internal address or person number) – which is only valid locally in each system – but through their owner (for example, the customer or the vendor). The key of the primary object is unique across all systems and used to determine the local address number in the target system: The address IDoc also contains information on the object type (for example, the customer) and the object key (for example, the customer number.)

This information is used by a callback function module of the application to determine the address number (or person number).

Message Types and IDoc Types of CAM

The following table provides an overview of the message types and the associated IDoc types for the three address types [Page 11]:

	Type 1	**Type 2**	**Type 3**
Message type	**ADRMAS**	**ADR2MAS**	**ADR3MAS**
Idoc type			
Release 4.5	**ADRMAS01**	**ADR2MAS01**	**ADR3MAS01**
Release 4.6	ADRMAS02 [Page 97]	ADR2MAS02 [Page 99]	ADR3MAS02 [Page 101]

Technical Details

The IDoc segments are generated together with the dependent structures (communication data and comments) from the BAPI structures for each address type. Naming is subject to the following rule: From structure **BAPI**<xxx>, the system generates structure **E1BP**<xxx>.
Since the IDoc segments are restricted to a length of 1000 bytes, multiple IDoc segment structures may be generated from one BAPI structure.

Examples:

From **BAPIAD1VL**, the system generates the structures **E1BPAD1VL** (fields up to a maximum total length of 1000 bytes) and **E1BPAD1VL1**.

From **BAPIADTEL**, the system generates **E1BPADTEL**.

From **BAPIADURI**, the system generates **E1BPADURI**, **E1BPADURI1** and **E1BPADURI2**.

IDoc Type `ADRMAS02` for Address Type 1

Segment	Meaning
E1ADRMAS	Header segment
E1BPAD1VL	BAPI structure for address type 1 distribution
E1BPAD1VL1	BAPI structure for address type 1 distribution (part 2)
E1BPADTEL	BAPI structure for telephone numbers
E1BPADFAX	BAPI structure for fax numbers
E1BPADTTX	BAPI structure for teletex numbers
E1BPADTLX	BAPI structure for telex numbers
E1BPADSMTP	BAPI structure for e-mail addresses
E1BPADRML	BAPI structure for remote mail addresses
E1BPADX400	BAPI structure for X.400 addresses
E1BPADRFC	BAPI structure for RFC addresses
E1BPADPRT	BAPI structure for PRT addresses
E1BPADSSF	BAPI structure for SSF addresses
E1BPADSSF1	BAPI structure for SSF addresses (part 2)
E1BPADSSF2	BAPI structure for SSF addresses (part 3)
E1BPADURI	BAPI structure for URI addresses
E1BPADURI1	BAPI structure for URI addresses (part 2)
E1BPADURI2	BAPI structure for URI addresses (part 3)
E1BPADPAG	BAPI structure for pager numbers
E1BPAD_REM	BAPI structure for comments on the address
E1BPCOMREM	BAPI structure for comments on the communication types

The elements of the header segment **E1ADRMAS** correspond to the import parameters of function module **BAPI_ADDRESSORG_SAVEREPLICA** and have the following meaning:

OBJ_TYPE	BOR object type of the address owner (for example, KNA1 for customer)
OBJ_ID	BOR object key of the address owner (for example, the customer number)
OBJ_ID_EXT	Extension of the BOR object key (GUID) (for example, for the SAP business partner)
CONTEXT	Semantic meaning of the address

These fields allow you to uniquely identify the address without specifying the ten-digit address number that is only valid locally in each system.

IDoc Type `ADR2MAS02` for Address Type 2

Segment	Meaning
E1ADR2MAS	Header segment
E1BPAD2VL	BAPI structure for address type 2 distribution
E1BPAD2VL1	BAPI structure for address type 2 distribution (part 2)
E1BPADTEL	BAPI structure for telephone numbers
E1BPADFAX	BAPI structure for fax numbers
E1BPADTTX	BAPI structure for teletex numbers
E1BPADTLX	BAPI structure for telex numbers
E1BPADSMTP	BAPI structure for e-mail addresses
E1BPADRML	BAPI structure for remote mail addresses
E1BPADX400	BAPI structure for X.400 addresses
E1BPADRFC	BAPI structure for RFC addresses
E1BPADPRT	BAPI structure for PRT addresses
E1BPADSSF	BAPI structure for SSF addresses
E1BPADSSF1	BAPI structure for SSF addresses (part 2)
E1BPADSSF2	BAPI structure for SSF addresses (part 3)
E1BPADURI	BAPI structure for URI addresses
E1BPADURI1	BAPI structure for URI addresses (part 2)
E1BPADURI2	BAPI structure for URI addresses (part 3)
E1BPADPAG	BAPI structure for pager numbers
E1BPAD_REM	BAPI structure for comments on the address
E1BPCOMREM	BAPI structure for comments on the communication types

The elements of the header segment **E2ADRMAS** correspond to the import parameters of function module **BAPI_ADDRESSPERS_SAVEREPLICA** and have the following meaning:

OBJ_TYPE	BOR object type of the person object (for example, BUS1006 for the business partner)
OBJ_ID	BOR object key of the person object (for example, the partner number for the business partner)
OBJ_ID_EXT	Extension of the BOR object key (GUID) (for example, for the business partner)
CONTEXT	Semantic meaning of the address

These fields allow you to uniquely identify the address without specifying the ten-digit address number and person number that are only valid locally in each system.

IDoc Type ADR3MAS for Address Type 3

Segment	Meaning
E1ADR3MAS	Header segment
E1BPAD3VL	BAPI structure for address type 3 distribution
E1BPADTEL	BAPI structure for telephone numbers
E1BPADFAX	BAPI structure for fax numbers
E1BPADTTX	BAPI structure for teletex numbers
E1BPADTLX	BAPI structure for telex numbers
E1BPADSMTP	BAPI structure for e-mail addresses
E1BPADRML	BAPI structure for remote mail addresses
E1BPADX400	BAPI structure for X.400 addresses
E1BPADRFC	BAPI structure for RFC addresses
E1BPADPRT	BAPI structure for PRT addresses
E1BPADSSF	BAPI structure for SSF addresses
E1BPADSSF1	BAPI structure for SSF addresses (part 2)
E1BPADSSF2	BAPI structure for SSF addresses (part 3)
E1BPADURI	BAPI structure for URI addresses
E1BPADURI1	BAPI structure for URI addresses (part 2)
E1BPADURI2	BAPI structure for URI addresses (part 3)
E1BPADPAG	BAPI structure for pager numbers
E1BPCOMREM	BAPI structure for comments on the communication types

The elements of the header segment **E3ADRMAS** correspond to the import parameters of function module **BAPI_ADDRCONTPART_SAVEREPLICA** and have the following meaning:

OBJ_TYPE_P	BOR object type of the person object (for example, BUS1006001 for the business partner employee)
OBJ_ID_P	BOR object key of the person object (for example, the partner number for the business partner)
OBJ_TYPE_C	BOR object type of the owner of the company address (for example, BUS1006 for the business partner)
OBJ_ID_C	BOR object key of the owner of the company address (for example, the partner number for the business partner)

IDoc Type ADR3MAS for Address Type 3

`OBJ_ID_EXT`	Extension of the BOR object key (GUID) (for example, for the business partner)
`CONTEXT`	Semantic meaning of the address

These fields allow you to uniquely identify the address without specifying the ten-digit address number and person number that are only valid locally in each system.

Process Flow of Distribution

Two perspectives exist for the data distribution process:

- The perspective of the sending system (Outbound Processing [Page 104]), and
- The perspective of the receiving system (Inbound Processing [Page 108])

Outbound Processing

Two procedures are available for distributing master data:

- Sending Data Directly [Page 105]
- Sending Data by Evaluating Change Pointers [Page 107]

See also Outbound Processing in Master Data Distribution [Ext.].

Sending Data Directly

In the source system, a function module of the application creates a master data IDoc during outbound processing (for the message type **DEBMAS** of the customer master, or for the message type **CREMAS** of the vendor master, for example) and passes it on to the ALE layer. In this case, the application must additionally initiate the creation of the dependent address IDoc. To do this, the application uses the following function modules of central address management which must be called as required:

- **MASTERIDOC_CREATE_REQ_ADRMAS** for company addresses
- **MASTERIDOC_CREATE_REQ_ADR2MAS** for personal addresses
- **MASTERIDOC_CREATE_REQ_ADR3MAS** for workplace addresses

Based on the object type and the object key, these function modules determine the local address number (person number) and fill the BAPI structures of CAM with the data for this number. So that the address number can be determined, the application must provide a callback function module with input parameters for the where-used-list and with the address/person number as output parameter, and add the function module to table **TSADRV** using transaction **SM30**.

For address type 1, the name of the callback function module must be entered in field **FUNC_AD1**. The template for the interface is function module **ADDR_BUS000_GET_ADDR1_KEY**.

For address type 2, the name of the callback function module must be entered in field **FUNC_AD2**. The template for the interface is function module **ADDR_BUS000_GET_ADDR2_KEY**.

For address type 3, the name of the callback function module must be entered in field **FUNC_AD3**. The template for the interface is function module **ADDR_BUS000_GET_ADDR3_KEY**.

The callback function modules defined that way are also used to determine the local address and person number for updates (SAVEREPLICA BAPIs) and other BAPIs of CAM (Change, GetDetail).

Once the above CAM function modules have filled the BAPI structures, the receiving systems are determined according to the ALE distribution model, and function modules **ALE_ADDRESSORG_SAVEREPLICA**, **ALE_ADDRESSPERS_SAVEREPLICA** and **ALE_ADDRCONTPART_SAVEREPLICA** are called depending on the address type. These function modules convert the BAPI structures into IDocs.

In order to determine the receiving systems in ALE Customizing, the CAM function module passes not only the object type (**BUS4001**, **BUS4002** or **BUS4003**) and the method (**SAVEREPLICA**) to the function module **ALE_ASYNC_BAPI_GET_RECEIVER**, but also a table with filter object values. Using these filter object values, an adequate link can be defined between the master object and the address IDoc in the distribution model. This is illustrated by the proposed model for customers and vendors (see also Settings in ALE Customizing [Page 109]).

The filter object values passed include:

Address type 1

AD_OBJTYPE	Object type of the owner

AD_OBJKEY	Object key of the owner
AD_CONTEXT	Semantic meaning of the address

As far as the special case of customer contacts (object type **BUS1006001**) is concerned, the object type (**AD_OBJTYPE2**) and object key (**AD_OBJKEY2**) of the object referenced by the contact (= customer) are also passed, since customer contacts do not have a distribution model of their own but are distributed together with the customer master. The values are determined using function module **WY_KNVK_REF_OBJTYPE_ID**.

Address type 2

The filter object values are determined in the same way as with address type 1.

Address type 3

AD_OBJTYPE	Object type of the owner
AD_OBJKEY	Object key of the owner
AD_CONTEXT	Semantic meaning of the address
AD_OBJTYP1	Object type of the higher-level object
AD_OBJKEY1	Object key of the higher-level object

So that possible message types can be assigned to the application (**DEBMAS** or **CREMAS**, for example) if the "Object key of the owner" filter element is selected during distribution model view definition, application developers must maintain the dependencies between the SAVEREPLICA methods and the message types in transaction **BD48**.

Sending Data by Evaluating Change Pointers

Address data to be distributed can not only be sent directly, but also by evaluating change pointers. In this case, the sending process is not triggered by the primary object (application object), but executed by the Shared Master Data (SMD) tool of the ALE layer. The SMD tool evaluates the change pointers (in a periodic job, for example) and calls function modules **`MASTERIDOC_CREATE_SMD_ADRMAS`**, **`MASTERIDOC_CREATE_SMD_ADR2MAS`** and **`MASTERIDOC_CREATE_SMD_ADR3MAS`** for the addresses.

To ensure that central address management generates change pointers when modifications are saved, the indicator for writing change pointers must be set for the master data object in table **`TSADRV`**. This is necessry for the customer and vendor master, for example. For bank master data, no change pointers are generated for the addresses since bank addresses are always sent directly.

In addition, writing change pointers must be active for the message types of CAM in ALE Customizing (see: Settings in ALE Customizing [Page 109]).

Inbound Processing

Once the address IDoc has run through the ALE and communication layer, it is available for inbound processing in the target system. The ALE layer calls the function module **`IDOC_INPUT_ADRMAS`** (or **`IDOC_INPUT_ADR2MAS`**, **`IDOC_INPUT_ADR3MAS`**) which converts the IDoc segments into BAPI structures again. Subsequently, method **`SAVEREPLICA`** is called for each address object (**`BAPI_ADDRESSORG_SAVEREPLICA`**, **`BAPI_ADDRESSPERS_SAVEREPLICA`**, **`BAPI_ADDRCONTPART_SAVEREPLICA`**). This method updates the addresses in the target system.

In ALE Customizing, you must ensure in advance that the address and the master object are sent to the same target system. To do this, you must define a dependency between the address message type and the master object message type in the distribution model. This ensures that distribution filters at master object level are also applied to the address object. For example, if you want to distribute only vendors for specific vendor numbers to a target system, then the addresses for other vendor numbers should not be distributed to this receiving system.

In ALE Customizing, you should additionally set up serialization between the message types of the master objects and address objects to define a specific order for creating, sending and updating the corresponding IDocs. Some address fields can be required fields for the application. The best way to achieve consistency in the target system is to ensure that addresses are processed by their primary object. If a new master object is created, the address information is then already available.

Comment

If an application object including address data is changed in the source system, the address number (person number) in the target system is determined as with outbound processing by the above-mentioned callback function modules of the application. If a new master object (including the address) is created in the source system, a new address number is assigned in the target system and stored together with the assignment information for the master object from the address IDoc in the temporary table **`ADOWNERREF`**. When the master data IDoc is processed, the new address number is stored as a reference to the address, and the corresponding record is deleted from table **`ADOWNERREF`**.

To do this, the application program must call function modules **`ADDR_ADOWNERREF_READ`** and **`ADDR_ADOWNERREF_DELETE`**.

Settings in ALE Customizing

You must perform the following activities in ALE Customizing to ensure that addresses are distributed together with the master objects:

1. Maintain distribution model:
IMG path: *Basis Components – Distribution (ALE) – Modelling and Implementing Business Processes – Maintain Distribution Model and Distribute Views*.
The best way to illustrate this activity is the proposed model delivered by SAP for customer and vendor data distribution. For details, see the corresponding chapter in the Implementation Guide under *Basis Components – Distribution (ALE) – Modelling and Implementing Business Processes – Predefined ALE Business Processes – Logistics – Master Data Distribution – Proposal for Distribution Model: Customer and Vendor Masters* (IMG path). The associated activity runs report **RWALEACU** which can also be used as a template for other objects with addresses.

2. Activate change pointer for each message type:
IMG path: *Basis Components – Distribution (ALE) – Modelling and Implementing Business Processes – Master Data Distribution – Replication of Modified Data*
In addition to activating the change pointers on a general basis in the current client, you must separately activate the change pointers for message types **ADRMAS**, **ADR2MAS** and **ADR3MAS**. This is not necessary if the data is to be sent directly.

3. Define serialization groups:
IMG path: *Basis Components – Distribution (ALE) - Modelling and Implementing Business Processes – Master Data Distribution – Serialization for Sending and Receiving Data – Serialization Using Message Types – Define Serialization Groups*.
SAP delivers the following serialization groups as proposals:

Serialization group	Processing order of message types
GRP_CRECOR_ADR	ADRMAS, CRECOR
GRP_CREMAS_ADR	ADRMAS, CREMAS
GRP_DEBCOR_ADR	ADRMAS, DEBCOR
GRP_DEBMAS_ADR	ADRMAS, ADR3MAS, ADR2MAS, DEBMAS

For objects other than customer and vendor, you must define and deliver adequate serialization groups with regard to the message types of the application. When you create a serialization group, you must specify a sequence number for each message type. These sequence numbers determine the order of processing.

As a rule, you must create the serialization groups both in the sending and in the receiving system (however, you define inbound processing [Page 108] only in the receiving system).

You can combine the sending and posting of IDocs that belong to a serialization group. To do this, choose *Basis Components – Distribution (ALE) – Modelling and Implementing Business Processes – Master Data Distribution – Serialization for Sending and Receiving Data – Serialization Using Message Types* in the IMG to model message type **SERDAT** and maintain the relevant settings such as the partner profile. For more information, see the IMG chapter

Serialization Using Message Types [Ext.] and its sections Maintain Distribution Model [Ext.] and Serialized Distribution Using Message Types [Ext.].

4. Define inbound processing:
   IMG path: *Basis Components – Distribution (ALE) – Modelling and Implementing Business Processes - Master Data Distribution – Serialization for Sending and Receiving Data – Serialization Using Message Types – Define Inbound Processing*.
   In the receiving system, you must make an entry for each combination of serialization group, message type, and sending system. These entries cannot be preconfigured and delivered by SAP since the sending system depends on the specific distribution model.

5. Define partner profile:
   IMG path: *Basis Components – Distribution (ALE) – Modelling and Implementing Business Processes– Partner Profiles and Time of Processing*.
   You can either generate the partner profiles from a completely maintained distribution model, maintain the partner profiles manually or manually edit partner profiles that have been generated.
   For partner type LS (= Logical system), you must maintain message types **ADRMAS**, **ADR2MAS** and **ADR3MAS** as the outbound parameters in the sending system for the partner number of the receiving system. In addition to the recipient port, you must specify the IDoc basic type. In the case of address type 1 distribution between two systems running the current release, you must also enter **ADRMAS02**.
   For serialization during outbound processing, select *Collect IDocs and transfer* as the output mode; otherwise, select *Transfer IDoc immediately*.
   In the receiving system, you must maintain message types **ADRMAS**, **ADR2MAS** and **ADR3MAS** as the inbound parameters for the partner number of the sending system. Enter BAPI as the process code. For serialization during inbound processing, select *Trigger by background program* in the *Processing* group box.

Data filtering during address distribution has not been implemented.

Addresses in User Administration

All processes of the SAP System are based on the user concept. A user can only log on to the system if a user master record with a password has been defined. To ensure the integrity of the business data in the SAP System, you can set up authorizations for transactions and restrict access to certain data. In the user master, you define and restrict the rights each user has in the SAP System.

The SAPoffice functions (Business Workplace) are automatically available to each user created in the system (the inbox, for example, and so on). This enables users to carry out their business and communication processes in a consistent work environment.

See also: Working with the Business Workplace [Ext.]

In the SAPoffice environment, users are referred to as SAPoffice or office users.

Integration of CAM

Mapping of User Addresses

Each user (or office user) has an address of the workplace address type [Page 11] (type 3). This address refers to a company address mapped in CAM by means of address type 1 (company address).

Data Conversion for User Addresses

As of Release 4.0, the address functions of user administration are implemented by CAM. Using the XPRA **RSXADR03**, the address tables of the users (**USR03**) and the SAPoffice users (**SADRP**) valid prior to Release 4.0 were migrated into the CAM tables together with the associated company addresses (**SADR**) (see: Data Conversion [Page 80]).

Tables converted

Source table	Target table(s)
USR03	ADRP, ADCP
SADR	ADRC
SADRP	ADRP, ADCP

During conversion, CAM creates where-used lists as follows:

- For converted user addresses, in table **ADRVP** (usage of the person)
- For converted company addresses assigned to users, in table **ADRV** (usage of the company address)

Once the conversion process is complete, both the user (through the fields **ADDRNUMBER** and **PERSNUMBER** of table **USR21**) and the SAPoffice user (**SOUD-USRADR**) point to the same address. They are both owners [Page 21] of this address.

All users are assigned to person group [Page 20] **BC01**. Similarly, CAM assigns address group [Page 20] **BC01** to all company addresses that are associated with users. Company addresses created after the data conversion process is complete are always assigned to the same address group (**BC01**), and the users can only be assigned to addresses from this address group.

All addresses in table **SADRP** that are not associated with a user are assigned to person group **SOEX** (*external SAPoffice communication partners*) during conversion.

Application Tables

Table	Comment
`USR02`	Logon data of the user
`USR03`	Obsolete. Former address data of the users (prior to Release 4.0)
`USR21`	Assignment of user name – address key (address and person number) (as of Release 4.0)
`USCOMPANY`	User company with address (as of Release 4.5)
`SOUD`	SAPoffice – user definition

Maintenance of the Address Data

You maintain user addresses in transaction `SU01` on the *Address* tab. This transaction contains a subscreen for address type 3. Additional functions for maintaining addresses are provided within transaction `SO12` (office user maintenance) where you choose the address icon to display an address dialog box, and similarly to `SU01` within transaction `SU3` (user profile).

Assignment of a Company Address

Each user in user administration has a workplace address [Page 11] and can therefore be assigned to a company address. You can maintain these addresses in transaction **SU01** by choosing *Environment → Maintain company address*.

User administration maps company addresses using the address type 1 (company addresses) of the CAM data model.

For workplace addresses, n:m relations can be defined between company addresses and persons (contacts) in CAM. This has been done for SAP business partners, for example. The data model of user administration, however, allows only 1:n relations. Therefore, a person cannot be assigned to multiple company addresses simultaneously. When you maintain user addresses, you can choose *Assign other company address*... or *Assign new company address*... to assign another company address (or user company) to the user. The current assignment to the company address is deleted.

Prior to Release 4.6

In Release 4.0, you can use the parametrized transaction **SADR** of central address management to maintain company addresses. As of Release 4.5, transaction **SUCOMP** is used.

In Release 4.0, company addresses for users are stored as addresses without owners [Page 21] with address group **BC01**.

As of Release 4.5, the primary object *user company* (table **USCOMPANY**) is available. XPRA **RSUSX001** is delivered for setting up the table **USCOMPANY** during a Release 4.0 to 4.5 upgrade.

System Behavior if Users are Deleted

The address of a meeting attendee, a mail sender or receiver can be displayed even if the corresponding user has already been deleted. To be able to do this, the system retains the **`SOUD`** record of the deleted user and sets the indicator **`SOUD-DELETED`** to **`'X'`**. The person group in the tables **`ADRP`** and **`ADRVP`** is changed from **`BC01`** to **`SODE`** (for SAPOffice Deleted). Person group **`SODE`** functions as a delete indicator. This means that although the address of a user is kept in the system when the user is deleted, it is never displayed in the hit list when a user searches for addresses.

Maintenance of Communication Data

When incoming mails are distributed, they are forwarded to those users who have maintained the corresponding address in their communication data.

For security reasons, users cannot maintain the entries for the *Internet mail* (**`INT`** and **`SMTP`**) and *remote mail* (**`RML`**) communication types when they maintain their own data. This data can only be changed in transaction **`SU01`** by users with administrator rights.

See also note 136186 in SAPNet.

Time Zones of Users and Associated Companies

When a user is created, the system proposes the time zone of the company address as the individual time zone. This is why the time zone is a required field when you maintain company addresses. If the company address assigned is changed and the new company address has a time zone which is different from the individual time zone of the user, the system issues a warning message and asks you if the individual time zone of the user should be adjusted.

Program Access to User Addresses

Any program access to user addresses requires usage of the BAPIs designed for this purpose, in particular **`BAPI_USER_GET_DETAIL`** and **`BAPI_USER_CHANGE`**.

Up to Release 4.5, these BAPIs contain the address data in a flat structure (**`ADDRESS LIKE BAPIADDR3`**). In addition, as of Release 4.6, all communication data of the address is contained in tables (for example, **`ADDTEL LIKE BAPIADTEL`**) in the interfaces. This makes it possible, for example, to use **`BAPI_USER_CLONE`** to distribute multiple email addresses while retaining the sequence number. This function is required to support distributed mail systems.

Distribution of Users with Their Addresses

Through the BOR object *user*, users are integrated into ALE distribution (central user administration). The users are distributed together with their user addresses. The associated company addresses (type 1) are distributed separately through BOR object **USRCOMPANY**. This makes it possible to maintain users and their address data in a central system and then pass this data on to other systems and clients. This eliminates the need to maintain users and addresses in different systems and clients.

For more information, see Users and Roles [Ext.]:

- Creating and Maintaining User Master Records [Ext.]
- Central User Administration [Ext.]
- User Distribution [Ext.]