Integration Scenarios in Central Address Management

Postal Validation

&

Duplicate Check and Error-Tolerant Search

Releases 4.6B/4.6C

TABLE OF CONTENTS

1. INTERFACES OF CENTRAL ADDRESS MANAGEMENT 3

2. POSTAL CHECK 3

2.1 SUBJECT AND PREREQUISITES 3
2.2 COMPONENTS OF THE INTERFACE 3
2.3 INTEGRATION OF THE BUSINESS ADD-IN ADDRESS_CHECK INTO THE ADDRESS CHECKS OF THE SAP SYSTEM 4
2.4 GENERAL RULES FOR HANDLING ERRORS AND MESSAGES 5
2.5 DESCRIPTION OF BUSINESS ADD-IN ADDRESS_CHECK 5
2.5.1 Method ADDRESS_POSTAL_CHECK 6
2.5.2 Method IS_ACTIVE_FOR_COUNTRY 6
2.6 REFERENCE STRUCTURE ADRC_STRUC 7
2.7 QUARTERLY ADJUSTMENT 8
2.7.1 DESCRIPTION OF PROCEDURE 8
2.7.2 TECHNICAL DETAILS FOR THE PARTNER REPORT /PARTNER/RSADRQU2 8
2.8 INTEGRATION WITH THE BUSINESS INFORMATION WAREHOUSE (BW) 10

3. DUPLICATE CHECK AND ERROR-TOLERANT SEARCH 12

3.1 ARCHITECTURE AND COMPONENTS OF THE INTERFACE 12
3.2 INFORMATION ON RELEASE LEVELS 14
3.3 THE THREE DATA TYPES OF CENTRAL ADDRESS MANAGEMENT (ADDRESS TYPES) 14
3.4 THE ADDRESS WHERE-USED LIST AND OWNER OBJECTS 14
3.5 GROUPING DATA INTO INDEX POOLS, CONTROL TABLES FOR THE SEARCH PROCESS 15
3.6 SAP CUSTOMIZING 15
3.7 DETERMINING THE FIELDS FOR ERROR-TOLERANT SEARCH 16
3.8 INTEGRATION OF THE BUSINESS ADD-IN ADDRESS_UPDATE INTO CENTRAL ADDRESS MANAGEMENT 16
3.9 INTEGRATION OF THE BUSINESS ADD-IN ADDRESS_SEARCH INTO APPLICATIONS IN THE SAP SYSTEM 17
3.9.1 Duplicate Check 17
3.9.2 Search Helps 18
3.9.3 Initial Setup 18
3.10 DESCRIPTION OF THE BUSINESS ADD-IN ADDRESS_UPDATE 18
3.10.1 ADDRESS1_SAVED 18
3.10.2 ADDRESS2_SAVED 19
3.10.3 ADDRESS3_SAVED 20
3.10.4 FINISHED 21
3.11 DESCRIPTION OF BUSINESS ADD-IN ADDRESS_SEARCH 21
3.11.1 ADDRESS_SEARCH 21
3.11.2 IS_COMPLETE 23
3.11.3 READ_INDEX_FIELD_LIST 23
3.12 OUTPUT OF (T100) MESSAGES BY THE PARTNER IMPLEMENTATION 23
3.13 INITIAL SETUP OF THE INDEX POOL FOR ERROR-TOLERANT SEARCH 24
3.14 VISUALIZING A HIT LIST AND READING OBJECT DATA 25
3.15 FAILURE OF THE PARTNER COMPONENT AND ASYNCHRONOUS PROCESSING 25
3.16 FILTERING 25
3.17 RESTRICTIONS 26

4. CUSTOMIZING 27

5. SAPNET NOTES 27

1. Interfaces of Central Address Management

Starting with R/3 Release 4.6B, the SAP System comes with open interfaces in central address management (CAM) that can be used by third-party products.

The interfaces were announced in the 4.6B release note "Duplicate and postal check interfaces".

This specification describes the interfaces for the postal validation, the duplicate check, and the error-tolerant search for addresses.

2. Postal Validation

2.1 Subject and Prerequisites

Starting with R/3 Release 4.6B, third parties can connect their external address check software to the SAP System. The purpose of this software is to ensure that the postal data is correct which is entered when addresses are maintained. Since Release 4.0B, the standard R/3 System has been delivered with a location file. This file is initially empty and must be filled by the customer. The partner software is designed to be used instead of or in combination with this location file. SAP provides interfaces (business add-ins) for this purpose; which can be used by the external programs.

Besides this functionality, a procedure has been developed which allows to update the address data stored in the SAP System after the regional structure data used by the partner tool as a basis for the checks have been updated in a quarterly adjustment.

In this context, the partner software must provide an additional tool in the form of a report (see below in Description of the Quarterly Adjustment).

Business add-ins constitute a Basis technology that allows you to efficiently manage the interface and any of its implementations, and provides an easy-to-use runtime environment.

A business add-in can be implemented in different software layers, for example, by international subsidiaries, Industrial Business Units (IBUs), partners, or customers. If a specific add-in does not allow multiple use (such as ADDRESS_CHECK for a country), only a single implementation can be active at a time for all software layers.

All technical requirements described in this specification apply both to partner implementations and business add-in implementations by other software layers (such as IBUs or customers).

You must always provide implementations for all methods of the interface for a business add-in.

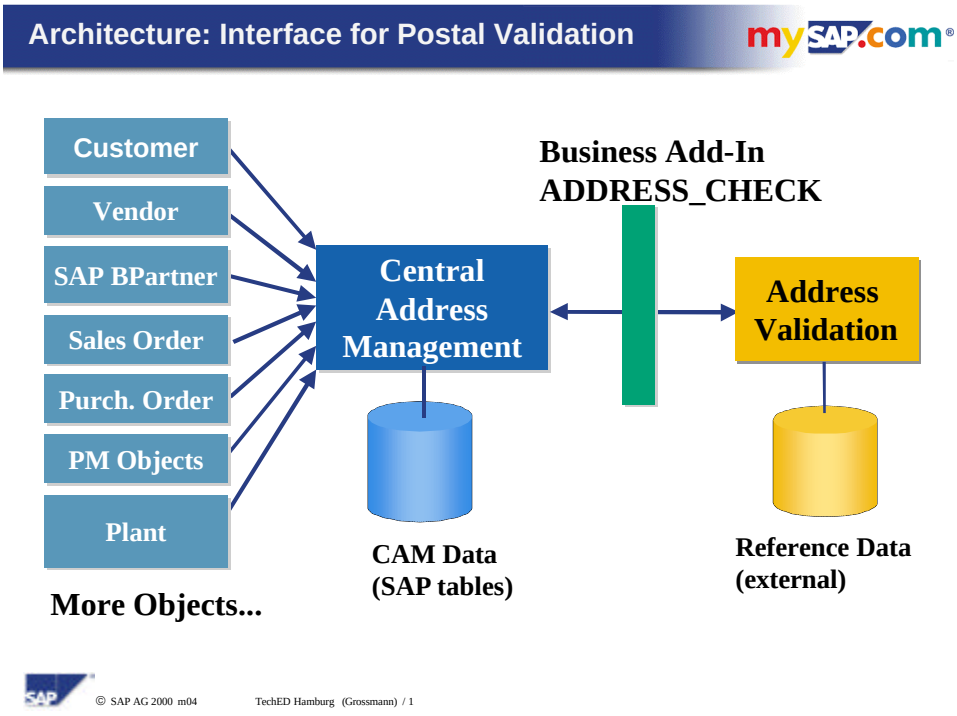
For a detailed description of business add-ins, refer to the SAP Library: *Basis Components® ABAP Workbench (BC-DWB) ®BC-ABAP Workbench: Tools; New Concepts®Business Add-Ins*.

2.2 Components of the Interface

The interface consists of the following two components:

- Business add-in ADDRESS_CHECK with the methods POSTAL_ADDRESS_CHECK and IS_ACTIVE_FOR_COUNTRY (see section 2.5)
- Report /PARTNER/RSADRQU2 for the quarterly adjustment (see section 2.7)

The following PowerPoint slide gives you a simplified overview of the interface architecture.



2.3 Integration of the Business Add-In ADDRESS_CHECK into the Address Checks of the SAP System

The function module ADDR_CHECK is used to check the contents of an address in central address management. This function module is called irrespective of the context (dialog, batch input, direct input, BAPI, function module without dialog) if an address is created or changed in central address management.

The isolation of the semantic checks from the dialog is based on the "Separation of Concerns" design principle and ensures that the same checks are performed in all program constellations.

In the function module ADDR_CHECK, a number of subsequent consistency checks are performed. The embedded call of function module ADDR_REGIONAL_DATA_CHECK is of central importance. This module groups together all checks for postal correctness.

In the function module ADDR_REGIONAL_DATA_CHECK, the following checks are called one after the other in the order specified below:

- Customer enhancement SZRS0003 (SMOD/CMOD), provided this customer enhancement has been activated in a CMOD project.
- Method ADDRESS_POSTAL_CHECK of business add-in ADDRESS_CHECK, provided an active implementation exists.
- Checks of the SAP regional structure using the SAP check tables, provided the location file for the current country has been activated.

Only one of these checks is normally activated for each country. Nevertheless, multiple checks can be enabled for a country which are then processed one after the other.

In all three checks, the structure ADRC_STRUC is passed and can be modified.

While the function module ADDR_CHECK is processed, the function module ADDR_POSTAL_CODE_CHECK is called before and after the function module ADDR_REGIONAL_DATA_CHECK is called. Function module ADDR_POSTAL_CODE_CHECK performs a number of plausibility checks for the postal code. For example, it checks its length and format. The rules for these checks are defined in Customizing in the country-specific settings. These checks are also available in previous releases (before the regional structure checks have been

implemented) and continue to be minimum checks that are executed even if no additional checks based on the SAP regional structure, or any partner or customer solution are active.

The purpose of calling ADDR_POSTAL_CODE_CHECK before ADDR_REGIONAL_DATA_CHECK is to ensure a minimum address consistency before stricter checks are performed.

The purpose of calling ADDR_POSTAL_CODE_CHECK after ADDR_REGIONAL_DATA_CHECK is to rule out any errors in the modified data which may be the result of wrong or incomplete reference data or errors in partner or customer implementations. This also supports the development and test of such solutions.

2.4 General Rules for Handling Errors and Messages

All messages (T100 messages) resulting from an address check are not issued directly with the message command but are collected and further processed in a message table (referred to as Error_Table).

This Error_Table is passed to the function module and evaluated based on the specific context. In dialog mode, for example, modules of central address management are normally responsible for displaying the messages using the ABAP command MESSAGE. In case of BAPI processing without a dialog, the messages are collected in the BAPI log. An important aspect when handling an address is whether a message of the "error" type has occurred.

Examples of errors:

- The postal code 79100 is not valid for Hamburg.
- The Trafalgar Square does not exist in Edinburgh.

When the first error occurs, the system terminates the check and rejects the address which then cannot be saved. In dialog mode, the user must first correct the address before it can be saved.

In the FIELDNAME field of the Error_table, you must always specify the name of the field to which a message refers. This field is used, for example, to correctly position the cursor in the dialog.

In the implementation of method ADDRESS_POSTAL_CHECK, the check must also be terminated after the first error occurs.

2.5 Description of Business Add-In ADDRESS_CHECK

This business add-in is designed to perform detailed checks for an address. It allows you to change or add address data in the process. For add-ins allowing data modification, the Basis technology does not allow multiple use (that is, multiple implementations), as it does, for example, for business add-in ADDRESS_UPDATE.

However, the business add-in ADDRESS_CHECK is filter-dependent. This means that a single active implementation may exist for each filter value.

The country of the address, that is, its ISO code, is used as the filter type.

Unlike the country key, which is defined by the customer and may differ from the SAP default, the ISO code of a country is always unique. As a result, external programs can query the value of this code. This filter makes it possible for a customer, for example, to simultaneously use different partner products in a global system for various countries.

The ISO codes are delivered by SAP as defaults in the country table. The customer needs properly maintained ISO codes for various checks in the system, including the address checks.

If a customer wants to use different implementations for addresses of a country, filtering is not the appropriate solution. In this case, the customer must create a business add-in implementation for the country in question which serves as a wrapper for several providers and contains only a dispatching function.

2.5.1 Method ADDRESS_POSTAL_CHECK

This method is used for the postal check and validation of the address data.

This method is an instance method.

The following parameters are passed:

CH_ADRC_STRUC	Changing parameter
This structure contains the fields of the postal address to be checked. You can add or correct data (see below, Reference Structure ADRC_STRUC).	
IM_DIALOG_ALLOWED	Import parameter
The business add-in is called as described above both in dialog mode and in processing without a dialog. If the dialog mode is used, you can add a user dialog which is displayed when the data is incomplete or wrong and allows the user to correct the address or enter the missing data. The parameter IM_DIALOG_ALLOWED specifies if the context is a dialog context in which an additional user dialog is allowed. When designing interface elements, (fehlt. you must) follow the guidelines contained in the SAP Style Guide to ensure that the user can work with a consistent interface. For tabular display and selection dialogs (for example, for cities, streets, postal codes), you are recommended to use the ABAP List Viewer (ALV).	
FLT_VAL	Import parameter
ISO code of the country of the address (ADRC_STRUC-COUNTRY). Since the filter values must be maintained as attributes of the implementation, the method is automatically called only if it is to be used for the country in question. Special case: A P.O. box address belongs to another country. Example: Address in Germany, P.O. box in Luxembourg. In this case, only the implementation valid for Germany is called as a result of the filtering. If the implementation which is active for Germany also provides a routine for Luxembourg, you can call this routine internally in the implementing layer.	
CH_T_ERROR_TABLE	Changing parameter
In this table, you must return all messages that occur when the address is checked. See above for the general message-related rules.	

2.5.2 Method IS_ACTIVE_FOR_COUNTRY

This method is used to determine if an extended postal check using the business add-in has been implemented for a country.

This method is a static method (class method).

The following parameters are passed:

FLT_VAL	Import parameter
ISO code of the country in the address (ADRC_STRUC-COUNTRY).	
CH_BOOLE	Changing parameter
This indicator must be set by the implementation if a check in method ADDRESS_POSTAL_CHECK is implemented for the country specified in the parameter FLT_VAL. Possible values are 'X' and SPACE.	

EX_SUPPRESS_CHECK_BEFORE	Export parameter
This indicator must be set by the implementation if the ADDR_POSTAL_CODE_CHECK function module call preceding the call of the business add-in ADDRESS_CHECK is to be suppressed for the following reason: The data in the business add-in should be checked and extended even if it does not yet comply with the formal criteria. Example: A dialog box appears in the implementation when the user enters a city name without a postal code, and the city name exists twice, for example, Frankfurt. To uniquely identify the Hessian city "Frankfurt am Main", it is sufficient to specify that the postal code begins with the number '6'. The actual postal code is then normally derived from the street and the house number. The user does not need to make further selections or even know the entire postal code. The function module ADDR_POSTAL_CODE_CHECK checks, however, if the postal code is five characters long if the address is an address within Germany. If only the number '6' is passed as the postal code, an error message is issued. To suppress such a check before the business add-in is called, method IS_ACTIVE_FOR_COUNTRY is used to determine if the implementation wants to disable the check. Possible values for the indicator are 'X' and SPACE.	

2.6 Reference Structure ADRC_STRUC

The meaning of most postal fields is described in the documentation in the Data Dictionary.

The fields HOUSE_NUM3, POSTALAREA, SORT_PHN, ADDRORIGIN, and ADDRESS_ID are not used.

In addition to the postal address attributes, the structure contains fields for coding the address.

These fields are filled when the SAP regional structure is used. Alternatively, they can also be filled by the implementing layer of the business add-in.

The fields are used to identify the city, street, and so on, on a language-independent level and are evaluated during the quarterly adjustment process, for example.

CITY_CODE	12-digit code for the city name
CITYP_CODE	8-digit code for the district name
STREETCODE	12-digit code for the street name
CITY_CODE2	12-digit code for the city name of a P.O. box
CITYH_CODE	12-digit code for the city name of the place of residence (different from the postal city)

The check status field plays a special role. The software layer performing the check (SAP regional structure, partner solution, or customer solution) uses this field to store the check status of the address.

This information is needed in the quarterly adjustment process and in evaluation reports.

Possible values are:

C	Address has been checked and conforms with the reference data (C means 'checked').
D	Address has been checked and does not conform with the reference data (D means 'differing').
SPACE	Address has not been checked.

The 'D' status makes sense if you want to save an address although it does not conform with the reference data currently used, for example, because the address belongs to a new district and is not yet contained in the reference data.

Only the user should be able to choose the 'D' status on an additional dialog box in dialog mode.

2.7 Quarterly Adjustment

In the context of the address check interface, the partner software must provide an additional tool in the form of a report for the quarterly adjustment.

This report must be created with the description /PARTNER/RSADRQU2 where you must replace /PARTNER/ with the actual namespace prefix. The extension RSADRQU2 is fixed and cannot be changed. The description is used in the SAP customer documentation on the interface functions. In the following sections, the report is always referred to as /PARTNER/RSADRQU2.

If IBUs or regional subsidiaries implement a solution, an appropriate analogous name should be used for the report in the corresponding namespace.

2.7.1 Description of Procedure

In the R/3 System, the report RSADRQU1 is called first. This report selects the addresses to be adjusted and temporarily stores them in a INDX-type table.

In the second step, the report /PARTNER/RSADRQU2 must be called. This report is to check the addresses stored temporarily based on the regional structure data changed, and update the data to reflect the status of the new data, if possible. If one of the addresses cannot be updated, it must be flagged accordingly.

Subsequently, the updated addresses should be stored in another INDX-type table. Function modules are available to read and save the addresses.

In the final step, the report RSADRQU3 is called which writes the updated addresses back to the SAP System.

2.7.2 Technical Details for the Partner Report /PARTNER/RSADRQU2

The report RSADRQU1 writes packages of addresses into an INDX-type structure. The maximum package size is specified by the administrator on the selection screen of the report.

Using the function module ADDR_READ_FROM_QU_INDX, you can read these packages separately.

You should do this in a loop over the package number which is passed to the function module in the parameter PACKAGE_NO, starting with '1'.

If the package number passed is identical to the number of packages that were generated using the report RSADRQU1, that means, if the package with the highest number has been read, the function module sets the parameter LAST_PACKAGE to 'X'.

Interface of the function module ADDR_READ_FROM_QU_INDX:

PACKAGE_NO	Import parameter
Package number	

LAST_PACKAGE	Export parameter
Indicator: Last package has been read	
PACKAGE_SIZE	Export parameter
Number of lines in the package read	

ADDRESS_TAB	Table parameter
Addresses read (export)	

PACKAGE_NOT_EXIST	Exception
The package specified does not exist.	
PARAMETER_ERROR	Exception
Error in the calling program (see system variables for the message).	
INTERNAL_ERROR	Exception
Fatal internal error (unexpected state, program error or data inconsistency).	

The function module returns the addresses in the table parameter ADDRESS_TAB. The number of lines in this table is returned in the parameter PACKAGE_SIZE and varies from package to package. The number can also be '0'.

The address data table has the structure of ADRC_QU1. The report /PARTNER/RSADRQU2 is to check the postal data contained in this table based on the general regional structure data changed and update the data to reflect the status of the new data, if necessary. If the data cannot be updated because the address data is unknown or incorrect, or because the new address data (such as a new street name or a new postal code) are not known, the addresses should be flagged accordingly (see below) but be nevertheless accepted.

The result must be written to a table with the structure of ADRC_QU2 together with the other data read. The structure ADRC_QU1 is contained in the structure ADRC_QU2 which allows the field contents to be copied directly. All data must be transferred, excluding, however, any changes that may result from an adjustment.

If an address cannot be adjusted for whatever reason, you must fill the field ERROR_FLAG in the ADRC_QU2-type structure with the value 'X'. In addition, you must fill the structure ADDR_ERROR included in ADRC_QU2 with information on the error occurred.

The errors are fatal errors if, for example, addresses of a country are passed for which no reference data exists. If the indicator ERROR_FLAG is set to the value 'X', the address in question is not taken into account by the report RSADRQU3.

If only the content check of an address against the reference data triggers an error or a warning, you must set ERROR_FLAG to the value SPACE. In addition, you must fill the message type ('E', 'W', 'I') and the message in the fields of the included structure ADDR_ERROR to allow the check result to be output in the log.

You must enter the check status for each address into the field CHCKST_QU of the ADRC_QU2-type structure based on the following scheme which considers the old check status of the addresses to be adjusted.

Old status in the field ADRC_QU1-CHCKSTATUS / New status in the field ADRC_QU2- CHCKST_QU	C	D	SPACE
C	Data is known and does not contain errors	Data is known and does not contain errors	Data is known and does not contain errors
D	May not be set	Data is unknown	May not be set
SPACE	Data is unknown	May not be set	Data is unknown

Similarly to how address data is read, adjusted data should be written in packages. It is possible to read and adjust multiple packages simultaneously and then pass them as one large package.

The function module ADDR_WRITE_TO_QU_INDX is available to save the address data adjusted.

Interface of function module ADDR_WRITE_TO_QU_INDX:

PACKAGE_NO_HIGH	Export parameter
Highest package number used in INDX	

ADDRESS_TAB	Table parameter
Addresses to be written (import)	

PARAMETER_ERROR	Exception
Error in the calling program (see system variables for the message).	
INTERNAL_ERROR	Exception
Fatal internal error (unexpected state, program error or data inconsistency).	

Using the table parameter ADDRESS_TAB which is based on the structure ADRC_QU2, a package of adjusted addresses is passed to the function module which is to be stored temporarily in the report RSADRQU3 for further processing. The package passed must contain at least one entry, that is, it must not be empty. Passing an empty package does not make sense and results in an error message.

The function module ADDR_WRITE_TO_QU_INDX divides the data into separate packages when writing it into the ADRC_QU2-type structure.

Having saved the data passed, the function module uses the parameter PACKAGE_NO_HIGH to return the total number of packages to be written into the INDX table that were created internally since the report /PARTNER/RSADRQU2 was started.

The field PCKG_NO_IN in the structure ADRC_QU2 is filled internally by the function module and is used for error analysis, for example. The field indicates during which function module call the address was passed.

If, for example, 20,000 addresses were passed in each of five calls, the function module can internally create 10 packages of 10,000 addresses each. PACKAGE_NO_HIGH finally contains the value 10, PCKG_NO_IN contains '1' for the first two 10,000 packages, '2' for the next two packages, and so on, up to 5.

2.8 Integration with the Business Information Warehouse (BW)

Starting with plug-in Release 2000, central address management is integrated with the Business Information Warehouse (BW) for SAP business partners.

In addition to the address data of business partner master records, the coded information on streets, cities and districts is also extracted from the OLTP system into the BW.

For the SAP regional structure, the contents of table ADRCITYT (texts for city keys), ADRSTREETT (texts for street keys) and ADRCITYPRT (postal districts) are extracted into corresponding info objects in the BW.

In order to transfer reference data of a partner software in the same way into the BW, it is sufficient to create an Excel file from the reference data in CSV (comma-separated) format and use this file as the data source for the BW.

The Excel file must have the table structure of the corresponding BW info object.

The corresponding info objects in the BW system are:

Cities: 0CITY_CODE Table /BI0/TCITY_CODE

Streets: 0STREET_COD Table /BI0/TSTREET_COD

Districts: 0CITYP_CODE Table /BI0/TCITYP_CODE

The structures have the following layout:

Transparent table **/BI0/TCITY_CODE** Active

Short description Texts: CAM characteristic: City code

Fields	**Key**	**Field type**	**Data type**	**Lgth.**	**Short text**
COUNTRY	x	/BI0/OICOUNTRY	CHAR	3	Country key
CITY_CODE	x	/BI0/OICITY_CODE	CHAR	12	CAM: City code
LANGU		LANGU	LANG	1	Language key
TXTMD		RSTXTMD	CHAR	40	Medium description

Transparent table **/BI0/TCITYP_CODE** Active

Short description Texts: CAM characteristic: City district code

Fields	**Key**	**Field type**	**Data type**	**Lgth.**	**Short description**
COUNTRY	x	/BI0/OICOUNTRY	CHAR	3	Country key
CITY_CODE	x	/BI0/OICITY_CODE	CHAR	12	CAM: City code
CITYP_CODE	x	/BI0/OICITYP_CODE	CHAR	8	CAM: City district code
LANGU		LANGU	LANG	1	Language key
TXTMD		RSTXTMD	CHAR	40	Medium description

Transparent table **/BI0/TSTREET_COD** Active

Short description Texts: CAM characteristic: Street code

Fields	**Key**	**Field type**	**Data type**	**Lgth.**	**Short description**
COUNTRY	x	/BI0/OICOUNTRY	CHAR	3	Country key
STREET_COD	x	/BI0/OISTREET_COD	CHAR	12	CAM: Street code
LANGU		LANGU	LANG	1	Language key
TXTSH		RSTXTSH	CHAR	20	Short description
TXTMD		RSTXTMD	CHAR	40	Medium description
TXTLG		RSTXTLG	CHAR	60	Long description

For more information, see the Data Dictionary in the BW System.

3. Duplicate Check and Error-Tolerant Search

3.1 Architecture and Components of the Interface

This specification describes the interface for the duplicate check and the error-tolerant search for addresses.

The interface consists of the following three components:

- Business add-in ADDRESS_SEARCH (see 3.11):
  Calls the error-tolerant search for addresses and the duplicate check.
- Business add-in ADDRESS_UPDATE (see 3.10): Using this interface for updating search indexes is a prerequisite for search functionality using business add-in ADDRESS_SEARCH. The business add-in ADDRESS_UPDATE can also be used in other contexts, for example, to respond to the saving of address data.
- Report /PARTNER/RSADRINI for initially setting up the search indexes (see 3.13). This report calls the standard function module ADDR_EXTRACT_FOR_DUPL_INDEX.

The error-tolerant search functions are currently integrated into the following master data. They can be used if an appropriate implementation of the interfaces is available.

- Customer master
- Vendor master
- SAP business partners

Business add-ins constitute a Basis technology that allows you to efficiently manage the interface and any of its implementations, and provides an easy-to-use runtime environment.

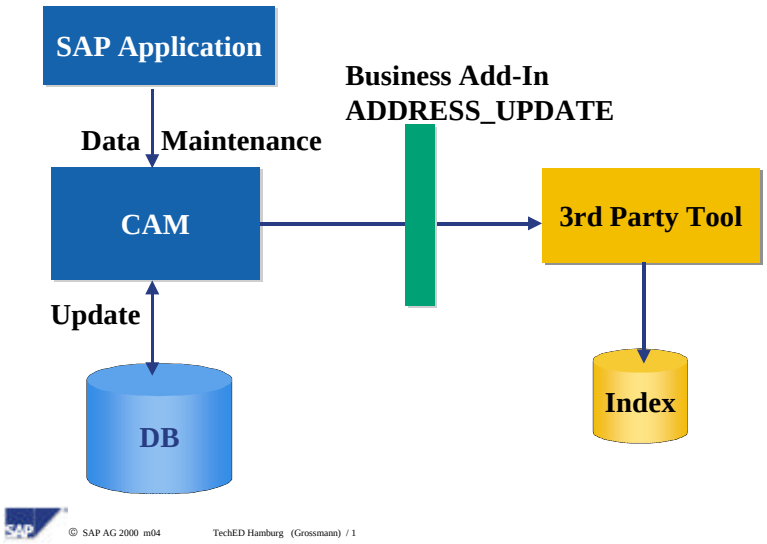
A business add-in can be implemented in different software layers, for example, by international subsidiaries, Industrial Business Units (IBUs), partners or customers.

All technical requirements described in this specification apply both to partner implementations and business add-in implementations by other software layers (such as IBUs or customers). To keep things simple, the term 'partner implementation' is always used in the following.

You must always provide implementations for all methods of the interface for a business add-in. For a detailed description of business add-ins, refer to the SAP Library: *Basis Components® ABAP Workbench (BC-DWB) ®BC-ABAP Workbench: Tools; New Concepts®Business Add-Ins*.

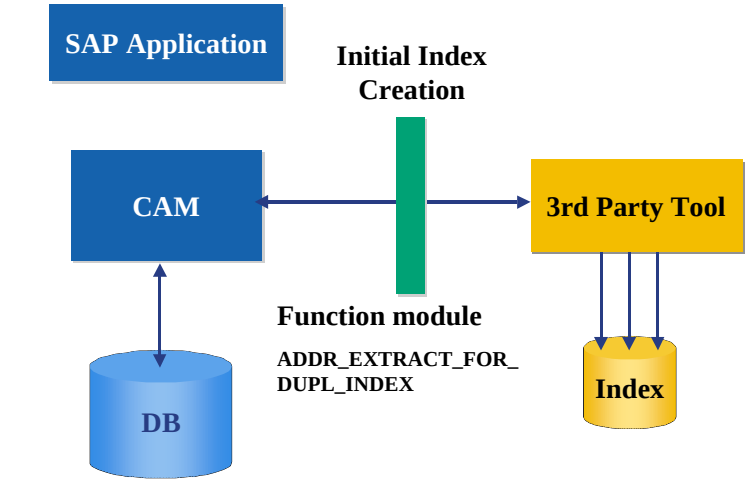
The following PowerPoint slides give you a simplified overview of the interface architecture.

Architecture: Index for Error-tolerant Search (1)

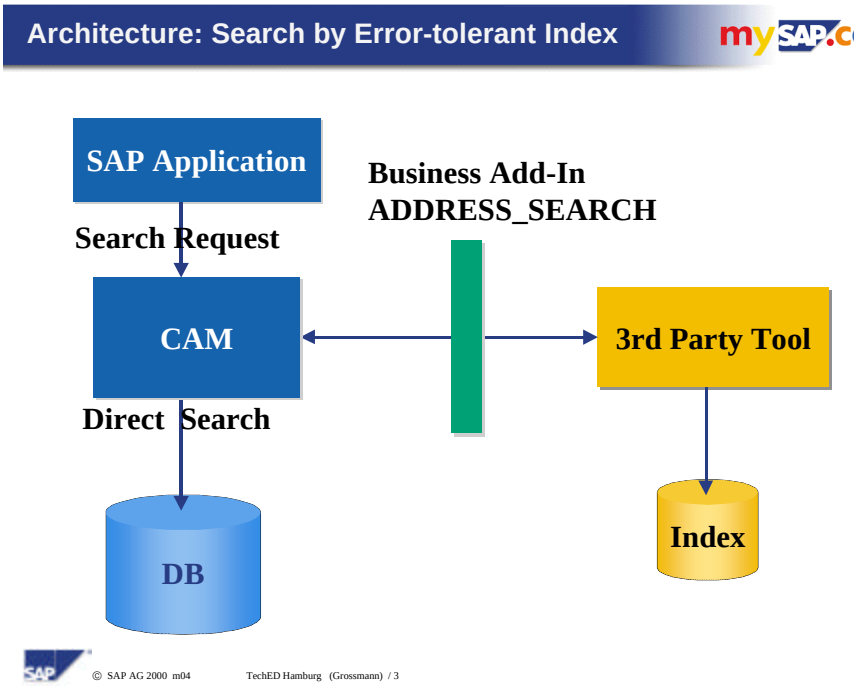


 TechED Hamburg (Grossmann) / 1

Architecture: Index for Error-tolerant Search (2)



 TechED Hamburg (Grossmann) / 2



Explanation: CAM = Central Address Management

3.2 Information on Release Levels

The interface specified here has been released since R/3 Release 4.6B. The interface is also available in the SAP CRM product. The information given below generally refers to the R/3 Release levels.

The R/3 Release Levels correspond to the following CRM Release levels:

The functionality of the interface in R/3 4.6B is the same as that in CRM 2.0A (note that CRM 2.0A is based on SAP Release 4.6A and that the 4.6B address functionality has been added to CRM 2.0A).

The functionality of the interface in R/3 4.6C is the same as that in CRM 2.0B (this corresponds to the delivery of CRM 2.0B based on SAP Release 4.6C).

3.3 The Three Data Types of Central Address Management (Address Types)

Central address management supports three data types:

- Type 1: Company addresses
- Type 2: Personal addresses
- Type 3: Workplace address

The three address types differ with regard to the attributes contained and the key components. For more information on these address types, see the documentation of function group SZA0, and the documentation on central address management in the SAP Library which is available starting with Release 4.6C.

3.4 The Address Where-Used List and Owner Objects

The where-used list of address management indicates which objects reference an address. A very good reference is the owner reference which belongs to the address owner. As far as master data is concerned, the owner of an address is the object by means of which the address can be maintained. This may be, for example, the customer or the vendor.

The where-used list allows you at any time to determine the owner of the address using the address number. For this reason, it is sufficient to store the address number (and the person number) as the identification of a record in the search index. You do not need to write ?fehlt: e.g.) the customer number to the search index.

3.5 Grouping Data into Index Pools, Control Tables for the Search Process

The search indexes for address data are basically client-specific as is the address data itself.

There can be further groupings within a client which are referred to as index pools.

The grouping of data indicates that there are different master data areas which differ largely with regard to their business integration into the SAP System and their technical implementation.

Currently, two such index pools are defined in the table TSADRVGRP. They have a technical key derived from the control table TSADRV:

KNA1 ADRNR	Customers, vendors, plants
BUT000 PARTNER	(Central) SAP business partner

It is possible to create further index pools. The standard system, however, contains only these two.

According to table TSADRVGRPE, the following single objects belong to the index pool KNA1 ADRNR:

KNA1 ADRNR	Customer master (address type 1)
LFA1 ADRNR	Vendor master (address type 1)
T001W ADRNR	Site master and plants (address type 1)
KNVK ADRNP_2	Consumer (natural person, address type 2)

These objects were grouped as an index pool because of overlappings between these areas.

A Retail site master, for example, can simultaneously be a customer, a plant, or a vendor. All three objects point to the same address. The 'consumer' (a special account group of the customer), on the other hand, is stored differently than a "normal" customer master from a technical point of view. As far as search aspects are concerned, however, there is no difference between these objects.

According to table TSADRVGRPE, the following single object is the only object that belongs to the index pool BUT000 PARTNER:

BUT000 PARTNER	SAP business partner

There are, however, two versions: one for address type 1, and one for address type 2.

It is important that both records of address type 2 (personal addresses with first name and last name) and address type 1 (company addresses with name1 and name2) are indexed in the two index pools and are evaluated together. In addition, the system should compare the name fields on a cross wise basis, that is, compare first name/first name and last name/last name as well as first name/last name and last name/first name (the same applies to name1/name2). If you want to find a business partner named Henry Miller, it does not matter if such a master record already exists in exactly that form or if only Henry Miller Ltd. exists in the system which could be the same business partner and should therefore be checked as a duplicate.

3.6 SAP Customizing

In the standard SAP Customizing, you must make settings for both index pools:

On the one hand, you specify on a general basis if the index pool should be active. On the other hand, you can define a percentage threshold (for example, 90,0) above which similar records should be regarded as a hit in searches without a dialog. Although you can set the percentage in the table, the percentage is not yet evaluated by the program in 4.6B and 4.6C.

The above settings are stored in the Customizing table TSADRVGRPC. You can maintain this table as of Release 4.6C in the Implementation Guide (IMG) by choosing Basis Components –> Basis Services –> Address Management. In Release 4.6B, you must directly call transaction SA17 to maintain this table.

3.7 Determining the Fields for Error-Tolerant Search

In Release 4.6B/C, you can index a defined set of address fields in the search index.

The following factors determine which fields will be included in the search index:

The control table TSAD10 lists all fields which can be indexed in the SAP System and for which this indexing process is supported by the standard programs. The setting depends on the address type.

Besides the fields entered in the table TSAD10 in 4.6B/C, other fields from additional address tables (such as email addresses) or application tables (such as date of birth or social security number of the business partner) will be supported as fields that can be indexed.

Normally, the features and functions of the partner tool determine which fields and how many fields can be indexed. With regard to the interface specified in this document, the set of indexed fields must always be a subset of the fields predefined by SAP. Besides, the partner should enable the customers to set the fields to be indexed for each index pool in Customizing. Since the partner may offer a restricted set of fields compared to the total set of fields supported by SAP, these Customizing settings are not provided in the SAP standard system.

The list of the fields for each index pool must be provided in the method READ_INDEX_FIELD_LIST of the business add-in ADDRESS_SEARCH.

If this method does not return a field list, a default setting is used which is delivered for each field and address type in the table TSAD10.

3.8 Integration of the Business Add-In ADDRESS_UPDATE into Central Address Management

The business add-in ADDRESS_UPDATE is a multiple use business add-in.

It provides all types of software layers with the capability to respond to changes made to address data.

It is not possible to change the data of central address management using exporting or changing parameters of the business add-in methods.

This business add-in is especially designed to update the search index for the duplicate check.

The methods ADDRESS1_SAVED, ADDRESS2_SAVED and ADDRESS3_SAVED of the business add-in are called within the function modules ADDR_MEMORY_SAVE and ADDR_SINGLE_SAVE. These function modules of central address management are called by the application program at the end of a transaction in order to save the CAM data to the database. Usually, only one of these function modules is called in an application transaction.

The address data of central address management can only be changed using these function modules. This ensures that consumers implementing the business add-in ADDRESS_UPDATE are informed of all changes made to CAM data.

While the function module ADDR_SINGLE_SAVE saves only one address at a time, the function module ADDR_MEMORY_SAVE saves all changes made while a transaction executes.

Since updating the search index usually involves a Remote Function Call, it is always better to update data in packages. To do this, you either need an indicator in the interface specifying that no further address changes will be passed, or a special method which is called at a later time.

The method FINISHED has been developed for this purpose. This method is called at the end of the two function modules ADDR_MEMORY_SAVE and ADDR_SINGLE_SAVE.

3.9 Integration of the Business Add-In ADDRESS_SEARCH into Applications in the SAP System

3.9.1 Duplicate Check

The duplicate check is integrated into the address maintenance dialogs for address types 1 and 2. To activate this functionality, the function module ADDR_ENABLE_DUPLICATE_CHECK is called in the application transactions.

The duplicate check has been activated in the following master data transaction:

- Customer master (R/3) including consumers (address type 2)
- Vendor master (R/3)
- SAP business partners (R/3, add-ons, and CRM).

During PAI processing of the address maintenance dialogs, all address data is initially checked for correctness, ideally also for complete postal correctness, either by means of the SAP regional structure or by means of a partner tool integrated using the business add-in ADDRESS_CHECK.

At the end of PAI processing, the method ADDRESS_SEARCH of the identically named business add-in ADDRESS_SEARCH is called for the duplicate check. The parameters defined by the application using the function module ADDR_ENABLE_DUPLICATE_CHECK are read with the function module ADDR_GET_DUPL_CHECK_PARAMETERS. In the application program, you can specify several object types as the search area, such as customer and consumer. Exactly one of these object types is identified as the dominant object type in the field MAIN_OBJ (which is ensured by a check in the function module ADDR_ENABLE_DUPLICATE_CHECK). Using the associated index pool, the system reads the Customizing table TSADRVGRPC and determines the field list using the method READ_INDEX_FIELD_LIST. The corresponding contents of the address fields are written to the transfer table IM_T_SEARCH_FIELDS.

The information as to which object type is dominant is also passed to the method ADDRESS_SEARCH in the table IM_T_OBJECT_TYPES. This information should be used by the partner tool, for example, to read the Customizing settings more specifically (for example, different settings for the customer master and for the vendor master), or to assign texts for messages and additional dialogs.

After all relevant parameters are determined, the method ADDRESS_SEARCH is called in which a search in the search index is to be performed.

It has to be decided which records are considered similar. It makes sense to determine a percentage for the degree of similarity. Since the SAP standard system does not yet allow customers to define the degree of similarity above which a records is regarded as similar, the partner implementation must offer the following features:

- The customer should be able to set the degree of similarity above which a record is considered similar, if possible, separately for customer, vendor, and SAP business partner. This value is called the threshold.
- An appropriate default must be predefined for this threshold which takes effect if the customer does not make any setting.

You must display the list of similar hits for IM_DIALOG_ALLOWED = 'X‘ using an additional dialog. You should allow users to select a record and choose it as a duplicate, to reject all hits returned as duplicates and continue processing, or to cancel the process since insufficient, wrong or too many search data could lead to a distorted search result.

When identifying a record, the address management system uses the where-used list of the address to determine the owner object of the address, for example, the customer number or the vendor number, from the address number. This information is passed back to the application program.

In the customer and the vendor master, if a duplicate has been identified, you can directly go to the maintenance of the other master record after confirming an additional dialog box. If you make this navigation step, the old record is discarded completely.

3.9.2 Search Helps

To enable applications to create search helps for use in the error-tolerant search, the call of the business add-in ADDR_SEARCH with search mode 'S‘ has been wrapped using the function module ADDR_FUZZY_SEARCH.

In Release 4.6B, this function module is used in the search help DEBIY for the customer master and KREDY for the vendor master.

To control which fields can be searched for in the index, the method READ_INDEX_FIELD_LIST of the business add-in ADDRESS_SEARCH is called. Fields that are not indexed are not offered for input on the selection screen for the search help.

3.9.3 Initial Setup

During the initial setup (see section 3.13n) the method READ_INDEX_FIELD_LIST of the business add-in ADDRESS_SEARCH is called to provide the fields required for the index in the interface of the function module ADDR_EXTRACT_FOR_DUPL_INDEX.

3.10 Description of the Business Add-In ADDRESS_UPDATE

One prerequisite for using the business add-in ADDRESS_SEARCH is the ongoing update of the search index by an implementation of the business add-in ADDRESS_UPDATE. The methods of ADDRESS_UPDATE were specified based on the three address types:

3.10.1 ADDRESS1_SAVED

IM_ADDRESS_NUMBER	Import parameter
Address number	
IM_T_OBJECT_TYPES	Import parameter
Table with object references. This table contains the owner object types of the address as they are stored in the control table TSADRV. Example: The following values are passed for the address of a customer master record: APPL_TABLE: KNA1 APPL_FIELD: ADRNR FLT_VALUE: (empty, of no significance in this example) MAIN_OBJ: (empty, of no significance in this example)	
IM_ADDRESS_GROUP	Import parameter
Address group to which the address is assigned (redundant information)	
IM_UPDATE_MODE	Import parameter
I = Insert U = Update D = Delete	
IM_SAVE_IN_UPDATE_TASK	Import parameter
Indicator which specifies if the changes are written to the database using "Call Function in Update Task". The processing type is determined by the application with which central address management is integrated.	
IM_WRITE_TO_SEARCH_INDEX	Import parameter

Indicator which specifies if the data is to be written to the search index for the duplicate check. This indicator is set if the address data belongs to an application object for which an entry exists in table TSADRVGRPE. **Exception:** An address of type 2 is created for the customer master in the "consumer" account group. However, an address of type 1 is created at the same time since KNA1 is filled with data including address data to ensure consistent evaluation. No additional entry must be created for this address type in the search index since, from a logical point of view, it is only an address.	
IM_T_XADRC	Import parameter
New version of table ADRC	
IM_T_YADRC	Import parameter
Old version of table ADRC	
IM_T_XADR2	Import parameter
New version of table ADR2	
IM_T_YADR2	Import parameter
Old version of table ADR2	

The remaining table parameters correspond to the other database tables for this address type.

3.10.2 ADDRESS2_SAVED

IM_ADDRESS_NUMBER	Import parameter
Address number	
IM_PERSON_NUMBER	Import parameter
Person number	
IM_T_OBJECT_TYPES	Import parameter
Table with object references. See above.	
IM_PERSON_GROUP	Import parameter
Person group to which the person belongs. For addresses of type 2, the person group is identical to the address group (redundant information).	
IM_UPDATE_MODE	Import parameter
I = Insert U = Update D = Delete	
IM_SAVE_IN_UPDATE_TASK	Import parameter
Indicator which specifies if the changes are written to the database using "Call Function in Update Task". The processing type is determined by the application with which central address management is integrated.	
IM_WRITE_TO_SEARCH_INDEX	Import parameter
Indicator which specifies if the data is to be written to the search index for the duplicate check. This indicator is set if the address data belongs to an application object for which an entry exists in table TSADRVGRPE.	
IM_T_XADRP	Import parameter
New version of table ADRP	
IM_T_YADRP	Import parameter
Old version of table ADRP	

IM_T_XADRC	Import parameter
New version of table ADRC	
IM_T_YADRC	Import parameter
Old version of table ADRC	
IM_T_XADR2	Import parameter
New version of table ADR2	
IM_T_YADR2	Import parameter
Old version of table ADR2	

The remaining table parameters correspond to the other database tables for this address type.

3.10.3 ADDRESS3_SAVED

IM_ADDRESS_NUMBER	Import parameter
Address number	
IM_PERSON_NUMBER	Import parameter
Person number	
IM_T_OBJECT_TYPES	Import parameter
Table with object references. See above.	
IM_PERSON_GROUP	Import parameter
Person group to which the person belongs (redundant information).	
IM_UPDATE_MODE	Import parameter
I = Insert U = Update D = Delete	
IM_SAVE_IN_UPDATE_TASK	Import parameter
Indicator which specifies if the changes are written to the database using "Call Function in Update Task". The processing type is determined by the application with which central address management is integrated.	
IM_WRITE_TO_SEARCH_INDEX	Import parameter
Indicator which specifies if the data is to be written to the search index for the duplicate check. This indicator is set if the address data belongs to an application object for which an entry exists in table TSADRVGRPE.	
IM_T_XADRP	Import parameter
New version of table ADRP	
IM_T_YADRP	Import parameter
Old version of table ADRP	
IM_T_XADRC	Import parameter
New version of table ADRC	
IM_T_YADRC	Import parameter
Old version of table ADRC	
IM_T_XADR2	Import parameter
New version of table ADR2	

IM_T_YADR2	Import parameter
Old version of table ADR2	

The remaining table parameters correspond to the other database tables for this address type.

3.10.4 FINISHED

IM_SAVED_ALL	Import parameter
Indicator which specifies that all address changes have been saved. The value is set to 'X' in function module ADDR_MEMORY_SAVE, and to SPACE in function module ADDR_SINGLE_SAVE.	
IM_ADDRESS_NUMBER	Import parameter
Address number (for single saving)	
IM_PERSON_NUMBER	Import parameter
Person number (for single saving)	
IM_ADDRESS_TYPE	Import parameter
Address type (for single saving)	

3.11 Description of Business Add-In ADDRESS_SEARCH

3.11.1 ADDRESS_SEARCH

Parameter description:

IM_SEARCH_MODE	Import parameter
Specifies the processing mode in which the search is called: 'I‘ = duplicate check during insert, 'U‘ = duplicate check during update, 'S‘ = search only.	
IM_DIALOG_ALLOWED	Import parameter
Indicator specifying that a dialog is allowed. The search can be called in dialog context or without a dialog, for example, in BAPIs. The standard dialog transactions (customer master, vendor master, and SAP business partners) expect the partner implementation to present an additional dialog as a dialog box that displays the potential duplicates or the hit list of the search. In some dialog transactions, the search result is to be evaluated on an application-specific basis. Despite the dialog environment, these transactions do not allow the partner implementation to present a dialog box. The indicator specifies if the partner is to use a dialog.	
IM_CURRENT_ADDRESS_KEY	Import parameter
Address key of the record processed (information for the duplicate check)	
IM_CURRENT_ADDRESS_TYPE	Import parameter
Address type of the record processed (information for the duplicate check)	
IM_SEARCH_IN_TYPE_1	Import parameter
Indicator: Search in addresses of type 1. Depending on the address type, the indicator specifies if the system is to find addresses of this type.	
IM_SEARCH_IN_TYPE_2	Import parameter
Indicator: Search in addresses of type 2	
IM_SEARCH_IN_TYPE_3	Import parameter
Indicator: Search in addresses of type 3	

IM_SEARCH_IN_ALL_OBJ_TYPES	Import parameter
Indicator: Search in all object types. This means that the system is to search in all index pools for all object types.	
IM_T_SEARCH_FIELDS	Import parameter
This table passes the selection criteria for the search fields. The structure of the table is generic since the fields to be indexed may come from different software layers (SAP Basis, SAP Application Basis or Core Application, SAP IBU Add-Ons). The values passed include the table name, the field name, and the field contents. The table name and the field name correspond to the database field names in table TSAD10.	
IM_T_OBJECT_TYPES	Import parameter
Object types in which the system is to search, normally one or more object types of an index pool. The table IM_T_OBJECT_TYPES can be neglected if the indicator IM_SEARCH_IN_ALL_OBJ_TYPES is set.	
IM_IGNORE_BUFFER	Import parameter
Indicator which specifies that the search is to be started independently of the last call. In dialog mode, each time data is released, it is processed and checked again in address management if changes have been made. In many cases, however, no changes are made to fields that are relevant to the search. For performance reasons, it is consequently not reasonable to search again in the search index using the same data. The partner implementation can hold the search criteria and, during the next call, can check if changes have been made to index-relevant fields. If you want to start an explicit new search (for example, by choosing a pushbutton on the dialog box), you must explicitly suppress the buffering. You use this parameter to do this.	
IM_THRESHOLD	Import parameter
Threshold for the check without dialog	
EX_NUMBER_OF_HITS	Export parameter
Number of hits	
EX_T_SEARCH_RESULT	Export parameter
Search result: The system must return all hits found during the search irrespective of whether a dialog has been processed in which the user flagged a hit as a duplicate.	
EX_SELECTED_ADDRESS_KEY	Export parameter
Address key of the record selected by the user.	
EX_SELECTED_ADDRESS_TYPE	Export parameter
Address type of the record selected by the user.	
EX_INDEX_INCOMPLETE	Export parameter
Indicator which specifies that the index pool is not in sync with the database. See the method IS_COMPLETE.	
EX_SEARCH_STATUS	Export parameter
Status of the duplicate check (user reaction). Possible values are the fixed values of the domain AD_DUPSTAT: 01 Duplicates/hits found 02 No duplicates/hits found 03 User identified no record as duplicate 04 Cancelled by user 05 No new check, search criteria unchanged 06 "Other value selection" function selected (for "search only" processing mode)	

Exceptions:

COMMUNICATION_ERROR	Exception

Communication error. External index search not possible. Potential reasons are: RFC connection not available, service not reachable, server down.	
INTERNAL_ERROR	Exception
Internal error. Internal index search not possible. Potential reasons are: program error, missing table settings, and so on.	
NO_RESULT	Exception
No search result because of inaccurate search criteria.	

The difference between return value 02 for EX_SEARCH_STATUS and exception NO_RESULT is as follows: If value 02 is returned, the system generally performed a search in the index. If the selection criteria are too inaccurate, for example, because just an asterisk (*) has been entered into a field, it does not make sense to start a complete search. In such cases, the partner implementation can trigger the exception NO_RESULT. The exact criteria may depend on the partner implementation or partner table Customizing.

3.11.2 IS_COMPLETE

Parameter description:

IM_INDEXPOOL	Import parameter
Logical index pool as specified in table TSADRVGRP (BUT000 PARTNER or KNA1 ADRNR)	
CH_BOOLE	Changing parameter
Indicator which specifies if the index pool is in sync with the R/3 database. If the server on which the index pool is stored is down for a while, the master records in the R/3 database may be temporarily out of sync with the index pool. This means that a newly added customer is not updated directly in the index. If this customer is entered again some time later, the duplicate index does not yet recognize the record as a potential duplicate, so that the record might be created twice. To recognize such situations and display appropriate warnings to the user, the method IS_COMPLETE should return for each index pool if the index is synchronous or if there are still records that need to be updated. To provide this information also at points other than searches, IS_COMPLETE was defined as a separate method.	

3.11.3 READ_INDEX_FIELD_LIST

You use this method to return the list of indexed field for each object type and address type.

Parameter description:

IM_OBJECT_TYPE	Import parameter
Logical object type. This type is determined by the Appl_table and Appl_field fields according to the entries in the table TSADRVGRPE. Example: LFA1 ADRNR is passed for the vendor master.	
IM_CURRENT_ADDRESS_TYPE	Import parameter
Current address type for which the field list is to be determined.	
EX_FIELD_LIST	Export parameter
List of indexed fields. The table names and field names must be passed as entered in table TSAD10.	

3.12 Output of (T100) Messages by the Partner Implementation

Errors and exceptions in the partner layer that occur when the address search is called must not be handled directly with the message command. Instead, you must either return an appropriate message using the exceptions of the method ADDRESS_SEARCH (for example, MESSAGE E123 RAISING COMMUNICATION_ERROR), or you must describe the status in the exporting parameters such as EX_INDEX_INCOMPLETE or EX_SEARCH_STATUS.

Depending on the context, the SAP standard programs ensure that the messages are, for example, sent as information messages or appropriately logged and that an adequate response to the return parameters is initiated.

3.13 Initial Setup of the Index Pool for Error-Tolerant Search

You must provide the report /PARTNER/RSADRINI for the initial setup of the index pool. This report calls the standard function module ADDR_EXTRACT_FOR_DUPL_INDEX in a loop.

You must replace /PARTNER/ with the actual namespace prefix. The extension RSADRINI is fixed and cannot be changed. The description is used in the SAP customer documentation on the interface functions. In the following sections, the report is always referred to as /PARTNER/RSADRINI.

If IBUs or regional subsidiaries implement a solution, an appropriate similar name should be used for the report in the corresponding namespace.

The function module has the following interface:

INDEX_POOL	Import parameter
Logical search pool to be set up (as specified in table TSADRVGRP)	
PACKAGE_SIZE	Import parameter
Package size for block selection (default: 10.000)	

ADDRESS_FIELD_LIST	Export parameter
Table with the contents of the fields to be indexed	
ADDRESS_OBJECT_LIST	Export parameter
Table with the object types for each address	
LAST_PACKAGE	Export parameter
Indicator: Last package read	

PARAMETER_ERROR	Exception
Error in the calling program (see system variables for the message).	
INTERNAL_ERROR	Exception
Fatal internal error (unexpected state, program error or data inconsistency).	

The function module ADDR_EXTRACT_FOR_DUPL_INDEX includes the following functions:

Using package processing (open cursor with hold, fetch next cursor), the function module reads all addresses into internal tables and subsequently passes the resulting tables ADDRESS_FIELD_LIST and ADDRESS_OBJECT_LIST to the calling program.

To determine the index-relevant fields, the method READ_INDEX_FIELD_LIST is called for the index pool specified. If this method does not return a result, the default value from the table TSAD10 is read.

The address where-used list is used to determine which addresses of the total data set are written to the index pool

The table TSADRVGRPE is used to read all object types for the index pool. All addresses belonging to these object types according to the where-used list are written to the index pool. All other addresses are not written to the index pool.

The corresponding field values for each address are passed in the structured table ADDRESS_FIELD_LIST. The field list has a generic structure since fields of other software layers that are to be determined dynamically may be contained in later releases.

The list of the owner object types for each address is passed in the table ADDRESS_OBJECT_LIST. The field FLT_VALUE is not filled with the key of the application object in this case (customer number, vendor number, business partner number) since this information should not be stored in the search index. Instead, this information must be determined after the search by means of the where-used list. Normally, only one object is the owner object (for example, customer: KNA1 ADRNR). In special cases, multiple objects can be owner objects (site master: customer + plant + possibly vendor). The information about the owner object type should be stored as a filter value for the indexed address to make it possible to effectively restrict the search results of a search (see section on filtering).

3.14 Visualizing a Hit List and Reading Object Data

You are recommended to use the Reuse tool ABAP List Viewer (ALV) for visualizing the hit list. This utility is described in the SAP Library. If you implement the dialog box without the ABAP List Viewer, you must absolutely observe the guidelines contained in the SAP Style Guide to ensure that the user interface is consistent.

When you display a hit list in the "search only" processing mode, you should provide the function for selecting additional values which is also present in the SAP standard search helps.

For performance reasons, a partner implementation should store the data in the search index only in the form required for an error-tolerant search. The implementation should not store the address data itself which is, for example, needed to visualize the hit list. This address data can always be read using the address number and person number keys.

You can use the function modules ADDR_GET_ARRAY, ADDR_PERSONAL_GET_ARRAY and ADDR_PERS_COMP_GET_ARRAY to read the address data for the partner dialog. To determine the owner object of the address, you can call the function module ADDR_REFERENCE_GET that returns all usages in table ADRV with the owners flagged as such (indicator ADDR_REF-OWNER in transfer table REFERENCE_TABLE).

The partner implementation should provide customers with a function in Customizing that allows them to set the selection and the sequence of the fields in the hit list depending on the object type (and the filter value, if required).

3.15 Failure of the Partner Component and Asynchronous Processing

If when the addresses are updated the search index cannot be updated simultaneously, processing must not terminate since this would restrict the availability of the master data maintenance function to an unacceptable degree. Since the business add-in ADDRESS_UPDATE is a multiple use add-in, it is not possible in the Workbench from a technical point of view to define exporting parameters in the interface. The multiple use feature is also the reason why no exceptions were defined for the methods of the business add-in ADDRESS_UPDATE. Exceptions that are triggered when the search index cannot be updated would not apply to all consumers but only to the methods for updating the duplicate index.

To keep the search index synchronous even if, for example, no RFC connection is available at the time of update, SAP suggests that the partner should perform a delayed update as soon as this is possible.

To do this, the partner component can, for example, hold all records that cannot be updated in a temporary table. Each time an address is saved, the partner can then attempt to update the records collected as well.

The status as to whether the search index is in sync with the SAP database is determined in the methods ADDRESS_SEARCH and IS_COMPLETE.

3.16 Filtering

As described above, a common index pool has been defined for the master data of customers and vendors. Any search access to this index pool potentially results in a mixed set of hits. To effectively restrict the results based on the object type, it makes sense to store the information about the object type with the record in the search index. For this purpose, the object type is passed in the business add-in ADDRESS_UPDATE and during the initial setup.

It is not possible to define different object types for SAP business partners.

There are plans for subsequent releases to write business partner roles, for example, as filter values to the index in addition to the object type. The field FLT_VALUE (filter value) is meant to be used for that purpose which is always passed in the same structure as the object type. When you implement the search index, you should already consider this type of filtering.

Role-based filtering for SAP business partners is to be used for performing error-tolerant searches for business partners specifically by tenants, vendors, customers, contract parties, and so on, without requiring a reselection in the R/3 database.

3.17 Restrictions

In Releases 4.6B/C, the duplicate check is only supported in dialog-based environments. In BAPIs and other function modules that do not use a dialog, this check is not yet performed.

In Releases 4.6B/C , the search/duplicate check for address type 3 (workplace address, contact) has not yet been integrated into the application transactions.

4. Customizing

In order to integrate Customizing activities of partner solutions, you must use the extension concept of the Implementation Guide (IMG).

You can find the description of the extension concept as follows:

Transaction SPRO_ADMIN

In the menu: Tools → Guideline: Extend SAP Reference IMG

The actual extension is defined in transaction S_IMG_EXTENSION.

The transport system might require a repair task.

However, the extensions can be imported into the customer system by means of a transport order of the partner and do not represent modifications to the customer system.

The structure containing the IMG activities for the partner solution should be inserted below the central address management node: Basis Components → Basis Services → Address Management.

The Customizing extension implemented that way provides the following benefits:

- The partner solution can use the structuring capabilities of the IMG.
- Partner Customizing is seamlessly integrated into the standard IMG. There is no need to use a separate Customizing transaction for the partner solution.
- Customers without an active implementation of the interface are not presented with IMG activities that are only required for existing implementations of the partner solution.

5. SAPNet Notes

The following SAPNet notes contain known errors and corrections for the interface described:

- 197803 Composite note: Error in interface for the address check
- 197217 Various errors in quarterly adjustment
- 196958 Composite note: Errors in CAM interfaces for duplicate check

The following note contains information on the interface for SAP customers:

- 176559 Interfaces of central address management