Open Catalog Interface (OCI)

Release 4.0

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help → General Information Classes and Information Classes for Business Information Warehouse* on the first page of the any version of *SAP Library*.

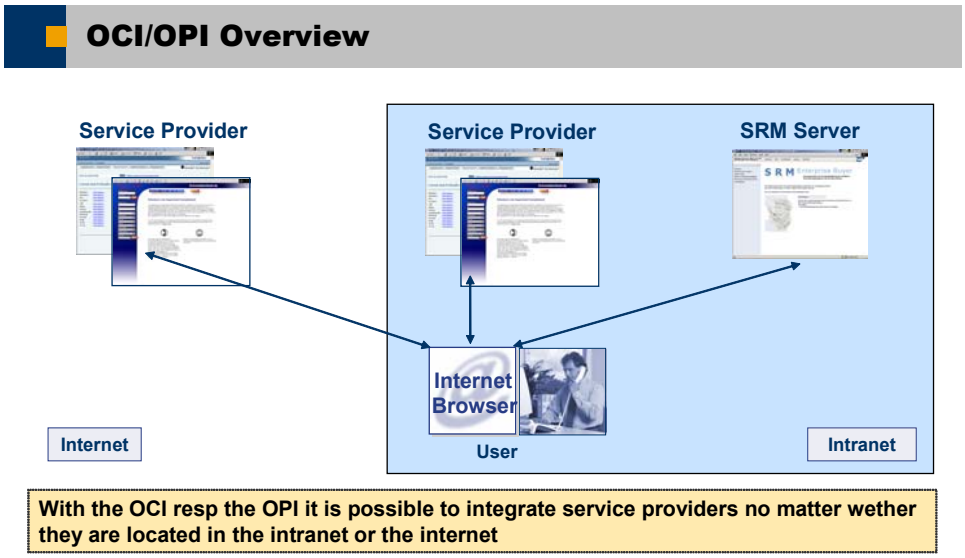
Typographic Conventions

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

1 Contents

2 Introduction

The *Open Catalog Interface* (*OCI*) incorporates external product catalogs into *SRM Server* applications. This way, data that is required in order to create shopping cart items in the *SRM Server* can be transferred directly from the external catalog to the *SRM Server* application. The interface uses the transfer mechanisms of Hyper Text Transfer Protocol (HTTP).



Graphic 1: System landscape

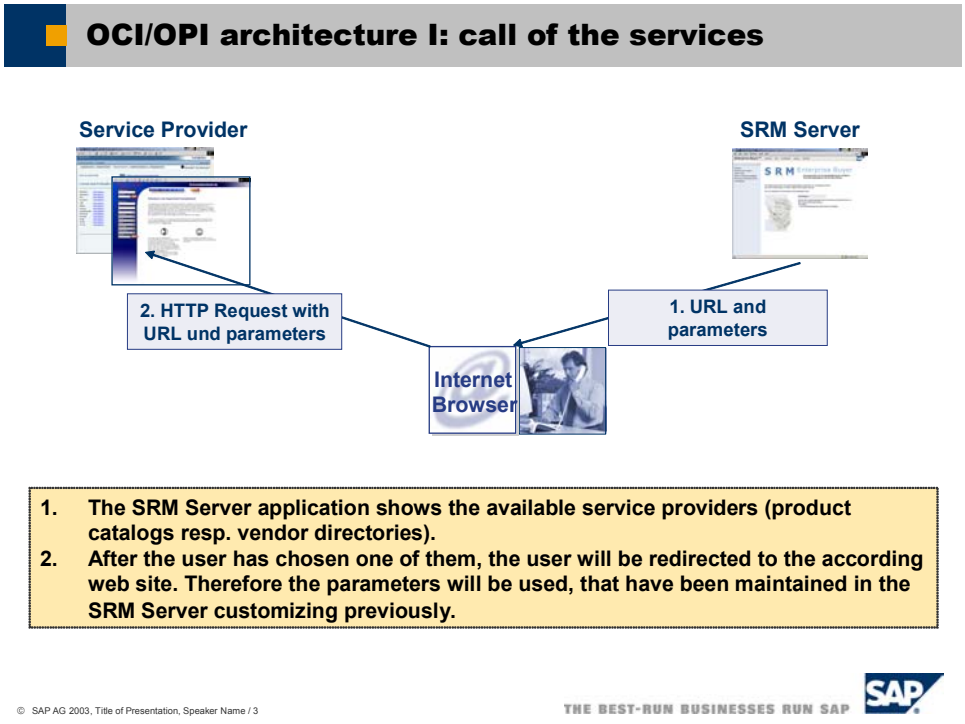
The user here is working with an *SRM Server* application, which displays the available catalogs. The user calls up one of these, selects the required products, and then transfers the product data back to the *SRM Server* application.

This documentation describes both the architecture and the structure of the *OCI* and thus provides all the information that is necessary to support the *OCI* with a product catalog. Also, the possible processes and their prerequisites are described. The documentation has been written both for producers of catalogs and for system administrators of *SRM Server* systems.

Section 3 describes how the *SRM Server* calls the catalog. Section 4 shows how the data is transferred from the catalog to the *SRM Server*. Section 5 covers the handling of the different browser windows and Section 6 gives some reference points for troubleshooting.

3 Calling Up the Catalog Using the SRM Server

In order for a product catalog to be called up via the Intranet or Internet, its URL must be known in the *SRM Server*. If the product catalog requires additional parameters for the call-up (for example, log-on names or language identifier), these must also be known in the *SRM Server* before the call-up.



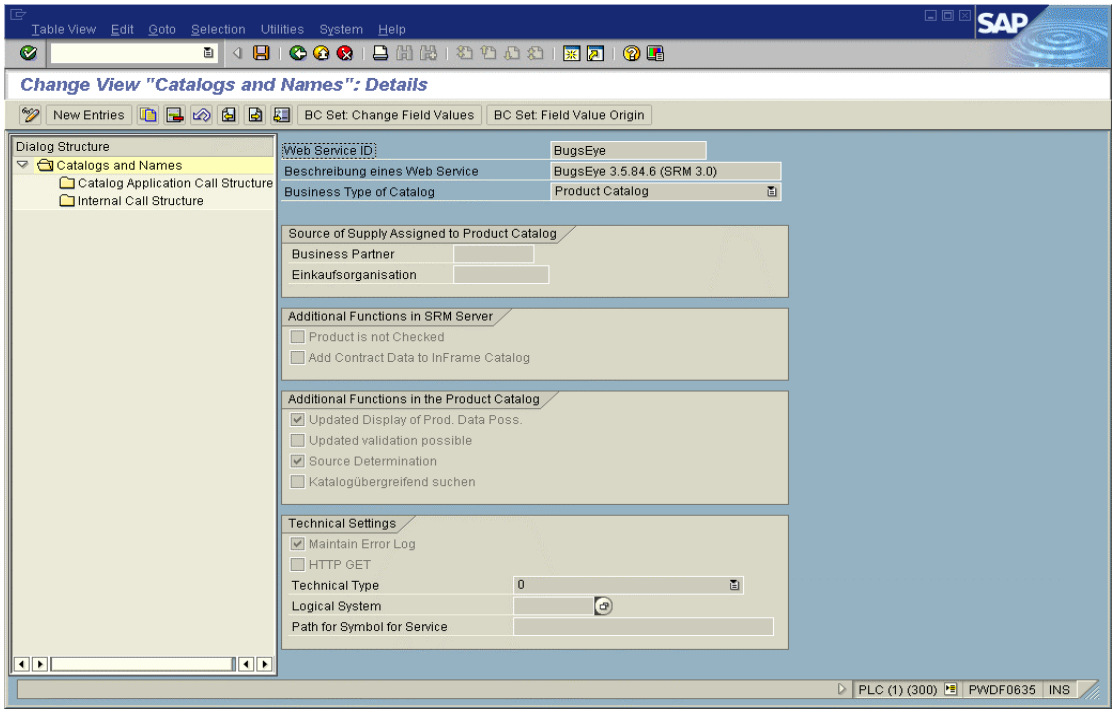
Graphic 2: Call-up

The URL and the catalog parameters are defined in the Implementation Guide (IMG) for *Supplier Relationship Management: SRM Server → Master Data → Define External Web Services (Catalogs, Vendor Lists etc.)*. In the technical settings (see graphic 3), you can set GET or POST as the HTTP method for the call-up, the standard value is POST. The user's browser is then used to call up the catalog using the URL and the parameters.

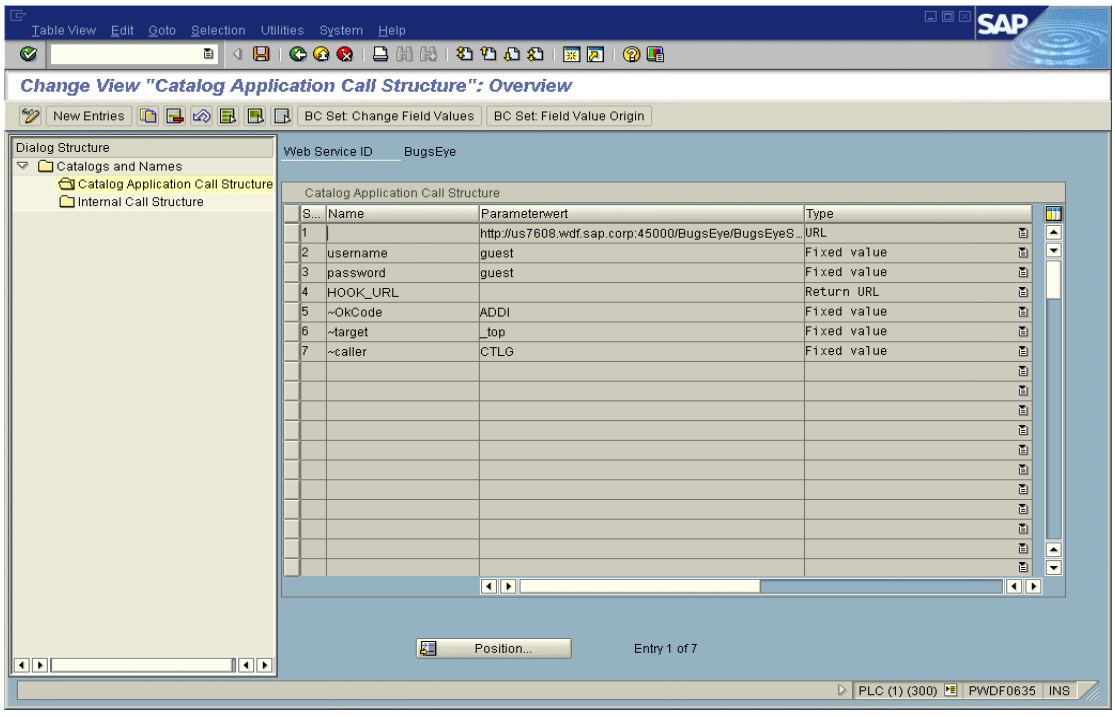
Depending on the situation, other parameters besides those defined in the Implementation Guide (IMG) for *Supplier Relationship Management* can also be transferred to the catalog on call-up. These might be, for example, to provide the catalog with generic data or to trigger specific functions in the catalog. A summary of the transferred parameters is shown in Section 3.3.

3.1 Settings in the SRM Server

The following two graphics show the maintenance screens for a product catalog in the *SRM Server*. You can find the complete documentation on these in the description of the IMG activity (*Define External Web Services (Catalogs, Vendor Lists etc.)*).



Graphic 3: The maintenance screen for general product catalog data



Graphic 4: The maintenance screen for the URL and parameters of the product catalog

In the *SRM Server*, you must enter the URL and parameters of the catalog in the following order in the IMG activity (*Define External Web Services (Catalogs, Vendor Lists etc.)*

3.1.1 URL of the Product Catalog

You enter the URL of the catalog in the first line of the call structure; all subsequent entries (including the return URL in line 4) are transferred to the catalog as parameters. You do not need to specify a parameter name for the URL itself. *URL* is used here as the type. If the URL is longer than the field, you can distribute the URL over several successive fields that are then all of type *URL*. Parameters may not be entered as part of the URL; they must be maintained separately as described in the following section.

3.1.2 Parameters

Following the URL, you must specify all parameters that the catalog requires on call-up. The provider of the catalog must have documented the names and valid values of these parameters.

The parameter type can be either *fixed value* or *SAP Field*. This parameter type determines how the value of the parameter is determined from the column *Content*. In the case of fixed values, the value of the parameter is entered directly in the column *Content*. In the case of parameters of type *SAP Field* (generic parameters), however, the name of a SAP System variable is there. For the value of the parameter, the content of this system variable at runtime is used. This way it is possible, for example, to transfer the system language as a parameter (you choose *SAP Field* as type and *sy-langu* as content). In this manner, you can transfer all the globally available fields at runtime from the *SRM Server* as parameters.

Transfer of additional parameters

If the fixed values and the generic parameter values are not sufficient for the chosen process, you can implement the Business Add-In (BadI) *Transfer Additional Parameters* (BBP_CAT_CALL_ENRICH) to transfer additional parameters from the *SRM Server* System to the catalog. In this BAdI, you can determine multiple name-value pairs that are then transferred to the catalog when it is called up. Also, if the format of generic parameters is to be changed (for example, if the system language is to be transferred in a different format: *DE-de* instead of simply *DE*), you can use this BAdI to convert these parameters.

As of *SRM* 2.0, a sample implementation is available for the BAdI *Transfer Additional Parameters* (BBP_CAT_CALL_ENRICH), which reads the relevant user data in the *SRM Server* and transfers it to the catalog (however, the sample implementation is only run if the business type of the category is set to *E-form*).

As an example of how this data can be used by the catalog, see the ASP page (ASP = Active Server Pages) that is stored in the *SRM Server* System in the Internet Service BBP_FREEFORM in the file *freeform.zip*. When you open all the data in an executable directory on an ASP-compatible Web server, you can use this page as a catalog application for ordering business cards. All the form fields automatically have the default values that have previously been transferred by the BAdI implementation. The user need only check the accuracy of the entries.

This ASP page can, of course, only serve as a template. The example is intended to show the options that are available on a project basis with a catalog and the implementation of the BAdI *Transfer Additional Parameters* (BBP_CAT_CALL_ENRICH).

3.1.3 Return URL

The return URL is required so that the data from the catalog can be transferred back to the *SRM Server* application via the user's browser. The catalog must place it in the *Action* attribute of the transfer form. The return URL is also transferred to the catalog as a parameter. It usually contains other parameters (see below) that must first be extracted and placed in separate input fields (of type *hidden*) of the form.

All parameters *after* the return URL are generated as parameters for the return URL; they are not transferred to the catalog as individual parameters (see graphic 4. lines 5-7). You can name the parameter for the return URL as you wish since the parameter must be evaluated by the catalog (usually *HOOK_URL* is used as the parameter name). The value of the return URL must be empty; the actual return URL is determined during run time.

As of *SRM Server 4.0*, the specification of the return URL and the following parameters (see graphic 4. lines 5-7) is optional in Customizing. If they are not specified, the return URL including the parameters is generated automatically. In this case *HOOK_URL* is used as the name for the return URL. The name of the return URL must be specified only if the catalog expects a different name for the return URL than *HOOK_URL*. Even if the name has been specified for the return URL, as of *SRM Server 4.0* the parameters for the return URL need no longer be specified; they are also generated automatically in this case.

3.2 Additional Functions in the Product Catalog

The product catalog can provide additional functions that can be used by *SRM Server* applications. In order to trigger these functions, additional parameters (defined within *OCI*) are transferred to the catalog when it is called up. If the product catalog supports one or more of the following functions, this is to be stated in the product catalog documentation. This is because an appropriate flag must be set in *SRM Server* Customizing so that the relevant *SRM Server* application can make use of the functionality. The fact that the interface supports the additional functionality does not necessarily mean that the functionality is actually used by a *SRM Server* application.

Currently the following functionality is supported by the *OCI*:

3.2.1 Detailed Display of a Product or Service

A product that has already been transferred to a *SRM Server* document via the *OCI* is to be displayed again later in the catalog. This serves to make available detail data that may not be defined in the *OCI* and that is also not available in the *SRM Server*.

In order to trigger the function, the following parameters are transferred to the product catalog on call-up:

Name	**Value**
FUNCTION	DETAIL
PRODUCTID	<Database key for the product in the catalog>

Then the product catalog should immediately display the detail view of the corresponding product. With this function no data is transferred from the product catalog to the *SRM Server*.

A prerequisite for this function is that previously, the first time this product was transferred from the product catalog to the *SRM Server* System, the field NEW_ITEM-EXT_PRODUCT_ID[] was filled with the database key of the product in the catalog. In addition, in *SRM Server* Customizing (in the IMG activity (*Define External Web Services, Catalogs, Vendor Lists etc.)* for the corresponding catalog, the flag *Display Prod. Data Again in Catalog* must be set.

3.2.2 Validation of a Product

If an item from a product catalog has been added to a template for an *SRM* document then it is possible, for example, that the price of the product has changed over time. If a new *SRM* document were to be created from this template, the price change would normally not be taken into account. The function serves to update the product information in the template while the *SRM* document is being created.

In order to trigger the function, the following parameters are transferred to the product catalog on call-up:

Name	Value
FUNCTION	VALIDATE
PRODUCTID	<Database key for the product in the catalog>
QUANTITY	<Current purchase order quantity>

The parameter QUANTITY is transferred as of *OCI 3.0* so that the catalog can determine the correct price from a scale, if appropriate.

Then the product catalog replies with an HTML page that contains a form with the product data in *OCI* format. The return URL must be split here as described in sections 3.1.3 and 4.1. The HTML page may not contain any visible elements (the input fields must be of the type *hidden*). The form must be sent automatically by JavaScript after the page has been loaded.

A prerequisite for this function is that previously, the first time this product was transferred from the product catalog to the *SRM Server* System, the field NEW_ITEM-EXT_PRODUCT_ID[] was filled with the database key of the product in the catalog. In addition, in *SRM Server* Customizing the flag *Validate Product Data from SAP Enterprise Buyer* must be set for the relevant catalog.

3.2.3 Sourcing/Product Search

In order to trigger the function, the following parameters are transferred to the product catalog on call-up:

Name	Value
FUNCTION	SOURCING
SEARCHSTRING	<Search term>
VENDOR	<Business partner number in EBP>

The catalog replies with the display of the search results that correspond to the specified parameters. By navigating further in the product catalog, you can select items as normal and transfer them to the *SRM Server* application.

This function has not yet been implemented in the *SRM Server.*

3.2.4 Background Search

In order to avoid the situation where a user searching for a particular product has to search multiple catalogs in sequence, a *Cross-Catalog Search* has been introduced as of *SRM 3.0*. The user enters a search term once in the *SRM Server* application and this term is transmitted to all catalogs that support the function when the catalogs are called up. The *SRM Server* displays the search results from all catalogs in a list and the user can select individual products from here.

In order to trigger the function, the following parameters are transferred to the product catalog on call-up:

Name	Value
FUNCTION	BACKGROUND_SEARCH
SEARCHSTRING	<Search term>

The product catalog replies with a HTML page that contains the search results in *OCI* format. All search results must be accessible on this page without having to use a scroll function

since the page is evaluated automatically. The form from this page may not be sent automatically to the return URL (in contrast to validation of a product).

A prerequisite for this function is that the flag *Search Cross-Catalog* is set in *SRM Server* Customizing for the relevant catalog.

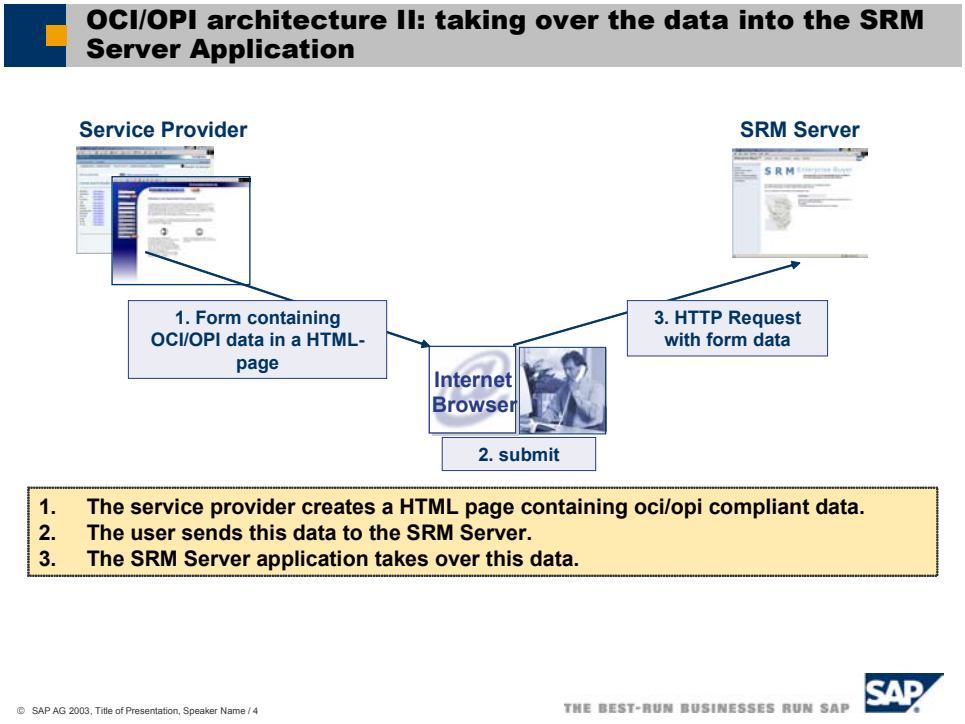
3.3 Overview of the Call-Up Parameters

The following parameters are transferred when the catalog is called up using either HTTP GET or POST, depending on what you defined in *SRM Server* Customizing:

Description	Name of the Parameter	Content or Example
Parameters from *SRM Server* Customizing	As specified in *SRM Server* Customizing	See Section 3.1.2
Parameters from a BAdI implementation	As implemented in the BAdI	See Section 3.1.2
Parameters to trigger additional functions	As described in Additional Functions in the Product Catalog	See Section 3.2
Return URL	If not specified differently in *SRM Server* Customizing: ‚HOOK_URL'	URL with parameters in the query string
Interface version *OCI*	OCI_VERSION	For example: 4.0
Interface version OPI	OPI_VERSION	For example: 1.0
Character set of *SRM Server* application	http_content_charset	For example: iso-8859-1
Target (HTML target) for return to the *SRM Server* application	returntarget	For example: _parent

4 Return From Catalog

A HTML form is used to transfer the selected product data to the *SRM Server*. This form is part of a HTML page that must be created by the catalog. This page (the last page that is displayed by the catalog) is sent to the user's browser. The user can now send the form from this page to the *SRM Server* application that then takes over the form data.



Graphic 5: Transfer of the data

In order that you can transfer the data from the catalog to the *SRM Server* via the user's browser, the return URL described in sections 3.1.3 and 4.1 must be inserted into the *Action* attribute of the HTML form created by the catalog (the *Action* attribute of the form is the URL to which the form data is sent).

The data to be transferred is transported within the input fields of the form; the field names must adhere to the syntax specified below. The *Type* attribute of the input fields should be *text* or *hidden*. The HTTP method recommended in all cases is POST because using GET can lead to browser-dependent length restrictions.

4.1 Use of the Call-Up Parameters From Section 3.3

- Use of the parameters from *SRM* Customizing is catalog-specific; here the only parameters transferred are those that the catalog expects.
- Use of the parameters from a possible BAdI implementation is project-specific; here the only parameters transferred are those that the catalog expects in the context of a customer project.
- The expected reactions to the parameters for triggering additional functions have already been described in section 3.2.
- The interface version has purely informative character; it clarifies which fields or processes are already supported by the *SRM Server* application.
- The return URL points to the current *SRM* application. It also contains further parameters. The URL without these parameters must be placed by the catalog into the *action* attribute of the transfer form. The parameter names and their values must be placed into the attributes *name* and *value* of the individual *input* fields of the transfer form; the fields are to be of the type *hidden*.
- The character set used by the *SRM* application is transferred to the catalog (parameter name: http_content_charset). When generating the HTML page with the transfer form, the catalog must use this character set and must insert it into the *meta* tag after the detail *charset=*.
- The target for the return to the *SRM Server* application (parameter name: *returntarget*) must be inserted into the *target* attribute of the transfer form.

4.2 Fields and Field Checks

The naming convention for the fields in the *OCI* is as follows:

`NEW_ITEM-<Field name>[<index>]`. The field type is always CHAR.

Field name	Length	Description
NEW_ITEM-DESCRIPTION[n]	40	Description of the item
NEW_ITEM-MATNR[n]	40	*SRM* product number of the item
NEW_ITEM-QUANTITY[n]	15	Item quantity (1.)
NEW_ITEM-UNIT[n]	3	Quantity unit for item quantity (3.)
NEW_ITEM-PRICE[n]	15	Price of an item per price unit (1.)
NEW_ITEM-CURRENCY[n]	5	Item currency (3.)
NEW_ITEM-PRICEUNIT[n]	5	Price unit of the item (if empty, 1 is used)(2.)
NEW_ITEM-LEADTIME[n]	5	Delivery time of the item in days (2.)
NEW_ITEM-LONGTEXT_n:132[]	∞	Long text for the item (4.)
NEW_ITEM-VENDOR[n]	10	*SRM* vendor number (business partner) for the item
NEW_ITEM-VENDORMAT[n]	40	Vendor product number for the item
NEW_ITEM-MANUFACTCODE[n]	10	*SRM* manufacturer number of the item
NEW_ITEM-MANUFACTMAT[n]	40	Item's manufacturer part number
NEW_ITEM-MATGROUP[n]	10	*SRM* material group for the item
NEW_ITEM-SERVICE[n]	1	Flag: the item is a service.

NEW_ITEM-CONTRACT[n]	10	*SRM* contract to which the item refers
NEW_ITEM-CONTRACT_ITEM[n]	5	Item within the *SRM* contract
NEW_ITEM-EXT_QUOTE_ID[n]	35	Number of an external bid for this item (as reference for a subsequent purchase order)
NEW_ITEM-EXT_QUOTE_ITEM[n]	10	Item of external bid
NEW_ITEM-EXT_PRODUCT_ID[n]	40	Unique database key for this item in the catalog
NEW_ITEM-ATTACHMENT[n]	255	URL of the attachment (the attachment must be accessible for downloading under this URL)
NEW_ITEM-ATTACHMENT_TITLE[n]	255	Title of the attachment (if this is empty the file name from the URL above is used)
NEW_ITEM-ATTACHMENT_PURPOSE[n]	1	Purpose of the attachment. *C* corresponds here to configuration.
NEW_ITEM-EXT_SCHEMA_TYPE[n]	10	Name of a schema via which it was imported in the *SRM Server*
NEW_ITEM-EXT_CATEGORY_ID[n]	60	Unique key for an external category from the schema above, **independent** of the version of the schema
NEW_ITEM-EXT_CATEGORY[n]	40	Unique key for an external category from the schema above, **dependent** on the version of the schema
NEW_ITEM-SLD_SYS_NAME[n]	60	Name of a system in the System Landscape Directory (SLD)
NEW_ITEM-CUST_FIELD1[n]	10	User-defined field
NEW_ITEM-CUST_FIELD2[n]	10	User-defined field
NEW_ITEM-CUST_FIELD3[n]	10	User-defined field
NEW_ITEM-CUST_FIELD4[n]	20	User-defined field
NEW_ITEM-CUST_FIELD5[n]	50	User-defined field

1.) 11 digits before the decimal point, 3 after it. Do not use commas for thousands.

2.) In whole numbers.

3.) Must be maintained as ISO code in the *SRM Server*.

4.) The field NEW_ITEM-LONGTEXT_n:132[] is an exception as far as the syntax of the index n is concerned. The field length is unlimited.

4.2.1 Required and Optional Fields

The following fields are required fields in all cases:

- Either NEW_ITEM-DESCRIPTION[n] or NEW_ITEM-MATNR[n] must be filled. Only one of the two should be filled.
- NEW_ITEM-QUANTITY[n]

The following fields are required fields depending on conditions:

- NEW_ITEM-UNIT[n] if NEW_ITEM-MATNR[n] has not been filled
- NEW_ITEM-CURRENCY[n] if NEW_ITEM-PRICE[n] has been filled
- NEW_ITEM-EXT_SCHEMA_TYPE[n] if NEW_ITEM-EXT_CATEGORY_ID[n] or NEW_ITEM-EXT_CATEGORY[n] are used

- NEW_ITEM-EXT_QUOTE_ID[n] if NEW_ITEM-EXT_QUOTE_ITEM[n] has been used
- NEW_ITEM-CONTRACT[n] if NEW_ITEM-CONTRACT_ITEM[n] has been used

All other fields are optional.

4.2.2 Product Numbers

There are four fields in the interface that describe product numbers:

- NEW_ITEM-MATNR[n]: The product number in the *SRM* System of the purchaser
- NEW_ITEM-VENDORMAT[n]: The vendor's product number
- NEW_ITEM-MANUFACTMAT[n]: The manufacturer's product number
- NEW_ITEM-EXT_PRODUCT_ID[n]: The number that uniquely identifies the product in the catalog.

These product numbers may not be mixed or used for other purposes; in particular the field NEW_ITEM-MATNR[n] may only be filled if the product number in the customer system is known to the catalog.

4.2.3 Configurable Products

Some products (such as PCs) can be configured in the catalog. However, the configuration information is not part of the *OCI* since the structure of this information differs greatly between providers. There are three alternatives for transferring such products with the *OCI* without losing the configuration information.

- The catalog can create a bid in the sales system and can store the configuration information there. It can then use the fields NEW_ITEM-EXT_QUOTE_ID[n] and NEW_ITEM-EXT_QUOTE_ITEM[n] to transfer a reference to the bid. The bid number is copied to the *SRM Server*. The configuration information is only available in the sales system if you use this alternative. This variant is suitable for the local and extended classic scenario since the bid reference is not transferred to MM backend systems as standard. If, however, you wish the bid reference to be transferred, you can copy it in BAdI BBP_CATALOG_TRANSFER into the purchase order text for the item.
- The field NEW_ITEM-LONGTEXT_n:132[] can be used to transfer the configuration information as text. The content of the field is included in the purchase order text of the *SRM Server* shopping cart and of the subsequent purchase order; this way the configuration information is available in the *SRM Server*.
- The fields NEW_ITEM-ATTACHMENT[n] and NEW_ITEM-ATTACHMENT_PURPOSE[n] can be used to transport the configuration information. Since you can transfer files of any type as attachments, you should ensure that the file can also be displayed (using proprietary or uncommon file types is therefore not recommended). If you use XML files, for example, you should ensure that the formatting information (XSLT) is also included so that the file can be displayed. The configuration information is also available in the *SRM Server* with this alternative. This variant is only suitable for the local and the extended classic scenario because attachments are not currently transferred to MM backend systems.

4.2.4 External Product Categories

As of *SAP Enterprise Buyer* 3.0, product categories from external schemas can also be used in the *OCI*. However, a prerequisite is the import of the relevant schema via the *Product Content Workbench* (*PCW*). If the imported schema is also used internally in the *SRM Server*, then no further action is required. If, however, there is to be a link from the product category from the external schema to an internal product category in the *SRM Server*, the mapping of the imported (or parked) schema must first be completed via the *PCW*. The fields NEW_ITEM-EXT_SCHEMA_TYPE[n] and NEW_ITEM-EXT_CATEGORY_ID[n] are used for the external product categories.

If, for example, an UNSPSC schema has been imported in the *SRM Server* under the name *UNSPSC_Vendor1*, then *UNSPSC_Vendor1* is the value that must be transferred in the field NEW_ITEM-EXT_SCHEMA_TYPE[n].

4.2.5 Customer-Specific Extensions in the OCI

To transfer additional fields in the *OCI*, there are two alternatives:

- **Using the Fields NEW_ITEM-CUSTFIELD1-5**

  These fields have no fixed semantics in the *OCI* and this means that you can transfer any contents in order to then evaluate them in BAdI BBP_CATALOG_TRANSFER. In Releases prior to *SAP Enterprise Buyer* 3.0 it is not possible to transfer these fields directly to the shopping cart item. If the contents are to be reusable later, they must be saved in the BAdI above in the customer's own database table.

  As of *SAP Enterprise Buyer* 3.0, it is possible to include these fields (without the prefix NEW_ITEM-) in the customer-specific Include CI_BBP_ITEM_SC; the contents are then automatically forwarded to the shopping cart and saved with it.

- **Using the Include CI_OCI_CUSTOMER_EXTENSION**

  This customer-specific Include is also available as of *SAP Enterprise Buyer* 3.0; it allows you to extend the *OCI* generically. All fields that are created in the Include can then be transferred by the catalog as *NEW_ITEM-<field name from Include>[n]*.

  The fields are also available in the BAdI BBP_CATALOG_TRANSFER. If the fields are also to be available in the shopping cart and are to be saved, they must also be created in the Include CI_BBP_ITEM_SC.

For more information on the use of customer-specific Includes in the *SRM Server*, see SAP Note 458591.

4.3 XML Variant of the OCI

The *OCI* can also process an XML file. Here the same architecture is used as in the pure HTML variant, this means the XML data is embedded in an HTML form for the transfer from the catalog to the *SRM Server* via the user's browser.

To transfer an XML file, besides the fields mentioned in section 3.3, two further HTML-*input* fields are used:

- **xmltype:**

  This parameter specifies the type of the XML file used so that the correct XML mapping can be found. Up to and including *SAP Enterprise Buyer 3.0* the mapping of the received XML data is done exclusively in the Business Connector which must be set up for this purpose for the relevant *SAP Enterprise Buyer* System.

  As of *SAP Enterprise Buyer* 3.5 the mapping can also be done in the *SRM Server* itself. A prerequisite is that the corresponding XML schema is used. The previous schemas are also supported.

  Valid values for the field type *xmlType* as of *SRM 3.0* are:

Value	Description	DTD/Schema/Mapping
ESAPO	Encoded SAP Object for *OCI* Version up to and including3.0	PDI_OCI.dtd/PDI_OCI.xsd/BC
ESAPO3.0	Encoded SAP Object for *OCI* Version up to and including 3.0	PDI_OCI_30.dtd/PDI_OCI_30.xsd/BC
ESAPO3.5	Encoded SAP Object for *OCI* Version as of 3.5	OpenCatalogInterface.xsd/im *SRM Server*.

- **~xmlDocument:**

  In this parameter the XML file that must correspond to one of the schemas above is transferred as a Base64-coded character set. The coding can be done either directly on the server page of the catalog or, as shown in the sample application, by JavaScript on the client's page.

5 Handling of the Browser Window

The catalog is normally displayed in a separate browser window. This window is both opened and closed again via the *SRM Server* application.

The HTTP response of the *SRM Server* application to the HTTP request with the *OCI* data consists of an HTML page that contains script for refreshing the window with the *SRM Server* application and for subsequently closing the catalog window. This page assumes that the JavaScript reference window.opener points to the browser window of the *SRM Server* application.

You may not change the name of the original window object because this would destroy the reference between the *SRM Server* application window and the catalog window. If this were to happen the *SRM Server* application would no longer be able to correctly display the data that has been transferred from the catalog or close the catalog window.

If the catalog opens additional windows during the search process it must close these before the data is transferred back to the *SRM Server*.

6 Troubleshooting

When you create purchasing documents using the *OCI*, errors can occur that are related to the meaning or validity of the data. In order to more easily identify such errors, you can activate the application log for a catalog linked via the *OCI* in *SRM Server* Customizing (for more information, see the Implementation Guide *Supplier Relationship Management → SRM Server → Master Data → Define External Web Services (Catalogs, Vendor Lists etc.) → Technical Settings → Maintain Error Log* see **Graphic 3***).*

You reproduce the error after you have switched on the application log. You can use transaction SLG1 to clear the application log in the *SRM Server*. Use BBP_OCI as the entry for the field *Object*. You can find more information on the application log in the F1 Help *Maintain Error Log*.