Cross Database Comparison
Enhancement Guide

Document Version 1.0 (2014-07-15)
MOTIVATION

Characteristics of generated CDC Data Extractors
With the Cross Database Comparison (CDC) tool, you define a graphical data model of both source systems and the mapping of the comparison fields between them. This data model offers a lot of flexibility, like joins on several tables and filters on fixed and dynamic criteria. Based on the data model, certain source types like ABAP and ADBC perform a generation, which is an automatic creation of the corresponding extractor ABAP function module or the native SQL statement. This generation happens based on fixed templates, to create the code to extract the data based on the object model, using best practices and common procedures to facilitate an “as-close-as-possible” match to the expected extraction scenario.

Limitation of generated CDC Data Extractors
However, there might be cases where the generated data extractor does not fully meet the requirement. Examples are:
- Instead of INNER JOINs you require other JOIN types
- Instead of JOINs, you rather require nested select loops, e.g. to have the possibility to insert additional logic between the selection of header and child tables
- You need to perform additional application logic, like special data conversion, key mapping, customizing lookup, extra filtering, and so on
- You require additional authorization checks
DATA EXTRACTORS FOR SOURCE TYPE ABAP

General Procedure to enhance the generated ABAP data extractor
- Create the data model (the source tables do not really need to match the real situation, but the mapping must consist exactly of all required key and data fields).
- Generate the ABAP RFC extractor function module.
- Optionally copy it to some other name.
- Using ABAP Workbench, manually change and enhance the coding.
  o Usually you should only modify the middle part with the database selections.
  o Please note that there is one selection to get the expected number of records, SELECT COUNT(*)..., and another part for the actual data extraction, so make sure they follow the same logic to get the same result.
  o Do not touch the general framework and the XML transformation.
  o Do not remove the authorization checks.
- In the CDC comparison instance, make sure you enter the correct function module name.

Example case
The actual goal of the comparison is quite simple: Compare the delivery numbers of all deliveries created yesterday between SAP EWM and SAP ERP, which have a successful goods movement for all items, to prove whether it was correctly replicated from EWM to ERP.

On the ERP side, the only enhancement is a relative date selection on all documents created yesterday, filtering on LIKP~ERDAT (simple date field).

On the EWM side, there were multiple extensions necessary, caused by the completely different data model:
- Relative date selection based on a timestamp field /SCDL/DB_PROCH_O~CRETST.
- Requirement to read all item’s status from /SCDL/DB_STATUS, because there is no appropriate status field on header level, together with an aggregation of header status telling all items have goods issue finished.
- Mapping of the EWM document ID to the corresponding ERP delivery number.

Note: As of ST 7.1 SP5, the relative date filtering is available in standard.

=> See appendix for code samples
DATA EXTRACTORS FOR SOURCE TYPE ADBC

General Procedure to enhance the generated ADBC SQL statement

- Create the data model (the source tables do not really need to match the real situation, but the mapping must consist exactly of all required key and data fields)
- Generate the native SQL counter and extractor statements
- Directly modify the SQL statement code on the CDC UI
  - To open the corresponding UI fields for input, you need to do a modification of the SAP-delivered source type for ADBC (or a copy of it in custom namespace), by following these steps:
    - Call transaction DSWP_CDC_SOURCES (“CDC: Customize Sources”).
    - Mark source type ADBC and select dialog structure “CDC Source Parameter”.
    - Switch to change mode.
    - See details for source type parameter SQL_COUNT (double-click).
    - Change “Param Usage in Obj.” from “Display only” to “Maintenance”.
    - Go back or press button “Next Entry”.
    - See details for source type parameter SQL_EXTR.
    - Change “Param Usage in Obj.” from “Display only” to “Maintenance”.
    - To save the changes you need a workbench request (cross-client customizing).
  - Make sure the selected fields in the SELECT-clause do really match the comparison object model as defined in the mapping information. The selected fields should exist as they are used to build a target structure.
  - Changing the FROM-clause is typically harmless, as long as the correct data is extracted at the end.
  - When changing the WHERE-clause, make sure the selection filters correctly match to the other source system.
  - Make sure you have an ORDER-BY-clause to force a correct sort order of the comparison key(s), in case the comparison key(s) is different to the table’s primary key.
- Attention: When you re-generate, the modified SQL statement will be overwritten!

=> See appendix for code samples
ENHANCEMENT SPOTS WITHIN CDC

The functionality of the CDC can be enhanced using two enhancement spots

- **DSWP_CDC_CONVERSION**
  This enhancement can be used to implement additional conversion IDs which can be entered in the mapping data and which can then be executed as part of the normal conversion based on conversion IDs.

- **DSWP_CDC_SOURCE**
  This enhancement can be used to modify the extracted data directly.

**Enhancement Spot DSWP_CDC_CONVERSION**

This enhancement spot can be used to modify method CONVERT_DATA of class CL_DSWP_CDC_CONVERSION. Method CONVERT_DATA is used to execute the different conversions based on Conversion ID as entered on the mapping detail screen.

**Procedure**

1. Create a new conversion ID
   
   Start transaction SM30 and enter view CDC_V_CONV_TYPE. Click on “Maintain” (Figure 1)

   ![Maintain Table Views: Initial Screen](image)

   **Figure 1**

   A maintenance dialog will be displayed. Click on “New entries” and enter the following data:
   
   a. **Conversion ID**
      
      The conversion ID is the technical name which will identify the conversion. It is a character field with length 10. The name has to start with “Y” or “Z” for customer specific conversions.
   
   b. **Field Label**
      
      This is the text that will appear in the drop-down list when selecting a conversion ID in the mapping detail screen:
2. Implement the coding

Start the class builder for class CL_DSWP_CDC_CONVERSION and navigate into method CONVERT_DATA. Click on button Enhance. Position the cursor on the line “ENHANCEMENT-POINT DSWP_CDC_CONV_ID SPOTS DSWP_CDC_CONVERSION” and perform a right click to display the context menu. Choose option Enhancement Operations → Create implementation. Enter a name for your enhancement in the displayed popup. Best practice would be to use the conversion ID as the technical name to have a simple reference between coding and definition.

Add the following coding to the new enhancement implementation

```
IF <fs_conversion>-conv_id EQ <conversion_id>.
   <YOUR CONVERSION CODE>
ENDIF.
```

Here <conversion_id> should be the technical name of the conversion ID you have created, while <YOUR CONVERSION CODE> contains your actual code. You can use the standard conversion routines as guidance how your conversion coding could look like.

Please implement SAP note 1942636 - CDC: Not possible to implement enhancements for data conversion if you have problems with the enhancement implementation.

**Enhancement Spot DSWP_CDC_SOURCE**

This enhancement spot uses BAdI DSWP_CDC_SOURCE_DATA and can be used to modify either the extracted XML-data directly or the data table generated from it. If you need to convert data because two mapped fields have different data types or formats and there is no suitable standard conversion ID, you
should rather create a custom conversion ID than implement the BAdI, in particular if the same conversion ID should be reused in the data models of multiple comparison objects. If you need more complex conversion like converting extracted data in one column based on the data in another column or if the conversion should only be used for the data model in a specific comparison object or even only for a specific comparison instance, you should rather implement the BAdI.

Filter possibilities for implementation
The following data can be used to determine the correct BAdI implementation
- **COMPARISON_INSTANCE**
  The technical name of the current Comparison Instance
- **COMPARISON_OBJECT**
  The technical name of the Comparison Object the current Comparison Instance is based on.
- **COMPARISON_SOURCE_NO**
  The number of the Source from where the data was extracted (1 or 2)
- **COMPARISON_SOURCE_TYPE**
  The current source type

Interface to be implemented
The class used to implement the BAdI has to implement interface IF_DSWP_CDC_SOURCE_DATA. This interface provides the following methods:
- **MODIFY_DATA_TABLE**
  This method may be used to modify the data table derived from the extracted data after transformation of the XML. The data is contained in changing parameter CT_SOURCE_TAB.
- **MODIFY_DATA_XML**
  This method may be used to modify the extracted XML data directly. The data is contained either in changing parameter CV_DATA_STRING or CV_DATA_XSTRING depending on whether the source type provides the XML in character string format or in binary string format. Importing parameter IV_XML_TYPE may be used to decide which import parameter should be used.

CREATING YOUR OWN SOURCE TYPES
Sometimes you might encounter the situation that a pure conversion of data is not sufficient but you need to create complete new access paths containing different connection parameter types not existing in the standard source types. In these situations you can create your own source types by defining first a source types including the needed and then implementing the interface IF_DSWP_CDC_SOURCE.

Procedure
**Definition of the source type**
Start transaction DSWP_CDC_SOURCES. You get a screen for view cluster maintenance. Click on button “New Entries”.

![Change View "CDC Source Types": Overview](image)

Enter the following data in the new screen
- Source Type: Technical key for the source type like ZTHS
• Description: A meaningful description for this source type like “Demonstration for enhancement”
• Implementing class: The class that implements this source type. This class has to implement interface IF_DSWP_CDC_SOURCE. For details see section “Creation of the class”. It is recommended that the name should contain the technical name, e.g. ZCL_ZTHS_SOURCE_TYPE.

• Web Dynpro component
  This fields contains the name of a Web Dynpro component to be used for parameter maintenance. If it is empty the default UI will be used. For details how to create your own component see section Embedding a specific UI.
• Execution modes: This defines how the source type may be invoked during comparison. The following possibilities exist:
  o Comparison can run in background or dialog
  o Comparison can run in dialog only
  o Comparison can run in background only
• SQL type for extract
  o Source type does not use SQL for extract
  o Source type uses ABAP OpenSQL for extraction from SAP DB
  o Source type uses native SQL for extraction from remote DB
• Data type extracted
  o g XML as character string (STRING)
  o y XML as byte string (XSTRING)
  o x Both, XML as character or byte string
• Data Dict. Support
  This flag is evaluated during data modeling to assess whether definitions of fields and tables are available for this Source Type. Based on this, value helps are offered and must be implemented.
• Iteration Support
  Sometimes you encounter situations where the CDC has to be executed while data exchange between the systems is ongoing. In those cases CDC will always detect temporary differences which might lead to false results. For these situations CDC can use an iterative mechanism which means that identified inconsistencies are extracted again based on the stored key information from the last result. This flag indicates whether your implementation supports this iteration.
• Versioning Support
  Set this flag if your implementation will invalidate the generated extractors after changes in the comparison object model. As examples, the standard source type ABAP will calculate a hash value from the object model and compares it with the one stored inside the generated extractor function module, to prevent execution of outdated coding, or the standard source type ADBC will delete the generated SQL statements.
• Single Source only
  If this flag is set the CDC will assume that the comparison is done by the extractor in the managed system and that instead of a block of data to be compared a comparison result will be returned.

Save your new definition. The next step is to define the needed connection parameters. To do this select your new Source Type and navigate in the tree on the left side to CDC Source Parameter.
As we create a new type, no parameters exist yet, but we have to create new ones. Click on “New Entries” to start this process. First enter the Source Type (in our case ZTHS) and a technical name for the parameter:

![New Entries: Details of Added Entries](image)

In addition, the following parameters define the behavior of the Source Type’s parameter:

- **Description**
  A description what this parameter will achieve and what it means. The description will appear as tooltip on the parameter in the UI.
- **Field Label**
  The label that will be used on the UI for the input field.
- **Parameter Sort Order**
  An integer number defining the order of display on the UI.
- **Parameter usage in Comparison Object**
  - **0 Invisible**
    Set this value if the parameter should be invisible in the Comparison Object UI.
  - **1 Display Only**
    Set this value if the parameter should not be changeable, e.g. because it is calculated depending on other parameters or values during generation.
  - **2 Maintenance**
    Set this value if the parameter should be maintained manually in the Comparison Object.
- **Parameter usage in Comparison Instance**
  - **0 Invisible**
    Set this value if the parameter should be invisible in the Comparison Object UI.
  - **1 Display only**
    Set this value if the parameter should not be changeable, e.g. because it is calculated depending on other parameters
  - **2 Maintenance**
    Set this value if the parameter should be maintained manually in the Comparison Instance.
  - **3 Copy Once from Object – Maintenance in Instance**
    The value of the parameter will be copied from the Comparison Object during initial creation of the instance, but can be overwritten manually in the Comparison Instance if this value is set.
4 Synchronize from Object – Maintenance in Instance
The value of the parameter will be copied from the Comparison Object during initial creation of the instance, but can be overwritten manually in the Comparison Instance if this value is set. In addition, the value can be synchronized again upon request with the Comparison Object.

5 Synchronize from Object – Display only in Instance
The value can be synchronized between Comparison Object and Comparison Instance but cannot be changed manually. The current value is displayed in the Comparison Instance.

6 Synchronize from Object – Invisible in Instance
The value can be synchronized between Comparison Object and Comparison Instance but cannot be changed manually. The current value is not displayed in the Comparison Instance.

- Parameter on result screen
  Should the parameter be displayed on result screen or not

- Mandatory in Comparison Object
  This flag controls whether the parameter is mandatory during Comparison Object maintenance

- Mandatory in Comparison Instance
  This flag controls whether the parameter is mandatory during Comparison Instance maintenance

- Authority Check
  Controls whether you want to execute an extra authority check based on the runtime content of the parameter

- Case-sensitive
  Determines whether the values are case sensitive or not, e.g. whether abc should differ from ABC. If the parameter is case insensitive the parameter’s values will be translated to upper case during saving at design time.

- Value help supported
  Controls whether the UI shall offer a value help, which must be implemented in the source type

- ACF file upload
  Set this flag if you need to access files on a local system. This will then activate the ACF file upload applet.

- Clear at Model change
  Controls whether the content of the parameter will be cleared after changes in the comparison object model

Once all parameters are maintained save the Source Type parameter definition and continue creating further parameters if needed.

Creation of the class
You start the creation of the class using the class builder via SE80 or SE24 directly. Next step would be to implement at least the interface IF_DSWP_CDC_SOURCE within this class and it has to be a subclass of class CL_DSWP_CDC_SOURCE. The interface provides attributes and methods that can be used to implement the interaction between the CDC UI and the managed system during definition and run time.

The interface provides the following attributes:

- GC_CDC_MESSAGE_CLASS
  This is a global constant containing the message class used in CDC.

- GC_MAX_VALUEHELP_ROWS
  This is a global constant containing the number of rows returned by the value helps.

- MX_CDC_EXCEPTION
  An attribute of type reference to exception class CX_DSWP_CDC_EXCEPTION used to handle class based exceptions.

- MT_PARAMETER
  This attribute contains the Source Type parameters and their values as name value pairs where the name is the technical name (ID) you provided during parameter definition. It is filled using method PARAMETER_SET.

The interface provides the following methods:
- **CHECK_EXTRACTOR**
  This method can be used to implement an existence check, e.g. if the generated extractors have to be transported between different systems to ensure they are available.

- **CHECK_VERSION**
  Here you can implement the versioning check if you have set the flag “Versioning Support” in the definition of the Source Type. It will be called at run time to check whether the extractor still fits to the version of the Comparison Object. If the version does not fit a corresponding exception has to be raised.

- **CONVERT_RELATIVE_DATE**
  Here you can implement the conversion of relative dates into specific date formats as needed by your system. If the needed format corresponds to the internal format as used in ABAP system you may use \texttt{cl_dswp_cdc Utils=>conv_dynamic_date_syntax} to perform this date transformation.

- **FETCH_BLOCK**
  This method will be called at run time to receive a new block of specified block size from the source system. Depending on whether you support iteration or not it also has to be able to use the comparison keys provided at run time.
  The extracted data has to be returned in the XML-structure needed for the comparison as character string or binary string where the tag-names correspond to the mapping names as defined in the corresponding comparison object.
  The XML has to be in \texttt{asXML} format, that means that in addition each row of data has to be encapsulated in \texttt{<item> ... </item>} and the complete table in \texttt{<table>..</table>} tags to allow the identification of the individual rows. An example where the mapping names “PARTNER”, “TYPE”, and “BPKIND” are used would look is displayed in Listing 1:

  ```xml
  <?xml version="1.0" encoding="utf-8"?>
  <asx:abap version="1.0" xmlns:asx="http://www.sap.com/abapxml">
    <asx:values>
      <TABLE>
        <item>
          <PARTNER>0000000001</PARTNER>
          <TYPE>1</TYPE>
          <BPKIND></BPKIND>
        </item>
        <item>
          <PARTNER>0000020000</PARTNER>
          <TYPE>2</TYPE>
          <BPKIND>0001</BPKIND>
        </item>
        ... 
        <item>
          <PARTNER>0000020008</PARTNER>
          <TYPE>2</TYPE>
          <BPKIND></BPKIND>
        </item>
      </TABLE>
    </asx:values>
  </asx:abap>
  
  Listing 1
  
  This transformation has either to be done locally in the managed system or centrally in the method itself based on a data table returned from the managed system. If the data table (for example \texttt{lt_source_data}) already contains the relevant mapping names as column names the conversion can be done easily in method FETCH_BLOCK in Solution Manager based on the following coding template (Listing 2)

  ```abap
  TRY.
  * Call transformation from ABAP internal table to XML string (char)
  IF ev_data_string IS REQUESTED.
  ```
CALL TRANSFORMATION id
SOURCE table = lt_source_data
RESULT XML ev_data_string.

* Call transformation from ABAP internal table to XML Xstring (binary)
ELSEIF ev_data_xstring IS REQUESTED.
   CALL TRANSFORMATION id
   SOURCE table = lt_source_data
   RESULT XML ev_data_xstring.
ENDIF.

* Catch cx_root as cx_transformation_error is only available as of
  basis version 7.0
CATCH cx_root INTO lx_root.
   lv_error_message = lx_root->get_text().
* Dummy message to populate SY-MSG* fields (accessible after RFC)
MESSAGE e001(00) WITH lv_error_message
   RAISING transformation_error.  "#EC_MG_PAR_CNT
ENDTRY.

Listing 2

- **FETCH_BLOCK**
  This method will be called during the actual comparison run to receive a block of data. The data will be provided as XML-document in string or xstring-format depending on the settings of the source types used in the comparison.

- **FETCH_COUNT**
  This method will be called initially at run time to determine how many records will roughly be compared. The returned value is also the basis for the progress indicator.

- **FETCH_COUNT_BOTH**
  This method will be called initially at run time to determine how many records will roughly be compared. The method “FETCH_COUNT_BOTH” will be called for single source types only (see section Definition of the source type for further details).

- **FETCH_RESULT**
  Method FETCH_RESULT will receive not a block of data to be compared but a complete comparison result from a managed system. It will be called for single source types only (see section Definition of the source type for further details).

- **GENERATE_EXTRACTOR**
  This method will be invoked when pressing the button “generate” in the Comparison Object’s UI. You can implement generations based on the entered parameters and data model like generation of function modules, SQL-statements etc.

- **GET_COLUMNS**
  The helper method GET_COLUMNS is used during design time as value help for field names (or equivalent) during data modeling. It will be called from method CL_DSWP_CDC_SOURCE=>VALUEHELP_COLUMNS.

- **GET_TABLES**
  This helper method is used during design time as value help for tables (or equivalent) during data modeling. It will be called from method CL_DSWP_CDC_SOURCE=>VALUEHELP_TABLES.

- **GET_VALUES**
  This helper method is used during design time as value help for field contents (or equivalent) during data modeling. It will be called from method CL_DSWP_CDC_SOURCE=>VALUEHELP_VALUES.

- **PARAMETER_SET**
  This method is used to set the parameter values as filled in the UI and should transfer them to the member table MT_PARAMETER

- **PARAMETER_VALIDATE**
  This method is invoked at design time during validation. You should check whether the entered parameter values are meaningful.

- **PARAMETER_VALUEHELP**
  This method can be used to implement a value help for the source type parameters. It receives the
technical name of the parameter and any filter data entered and has to return a table containing the actual allowed values and a description for each value.

- **PERFORM_PING**
  This method will be called for a connection test to the Source System and should assess whether the data source is reachable.

**Which methods have to be implemented?**

In general, the methods to be implemented depend on the intended behavior of your source type. The main criteria is whether Solution Manager should do the actual comparison using blocks of data extracted from the systems to be compared or whether the comparison should be done in the managed system and only a result should be transferred as determined by flag “single source only”. In the first use case you have to implement methods FETCH_BLOCK and FETCH_COUNT. In the second use case methods FETCH_RESULT and FETCH_COUNT_BOTH have to be implemented. You always have to implement one of these sets of methods. In both cases method PARAMETER_SET has to be implemented to provide the parameter values maintained in the UI to your class.

Besides this general decision, you have to decide which additional features your implementation should support. In case an extraction in multiple blocks should be supported for the usual use case, you have to return the concatenated key of the last entry in a block in field EV_ACT_KEY in addition to the block. This key will be provided during the next call of method FETCH_BLOCK to provide the offset in field IV_PREV_KEY. The delimiter to be used is contained in constant cl_dswp_cdc_compare=>gc_combined_key_separator_db. Your implementation of this method has to ensure that the returned entries are sorted according to the key across all blocks returned.

If iteration shall be supported, CDC will send a list of all keys for inconsistencies identified in a previous iteration to this method using parameter IT_FILTER_KEY. Each line of this table contains the key for an inconsistency where the individual keys are separated by constant cl_dswp_cdc_compare=>gc_combined_key_separator_db. You need then to perform an extraction for each of these keys and transfer the result back to CDC for comparison.

If your source type should support F4-helps for parameters, method PARAMETER_VALUEHELP has to be implemented. In addition, you can implement method PARAMETER_VALIDATE.

In case your source type implementation should support proposal of fields and table names you have to implement the following methods:

- **GET_COLUMNS**
- **GET_TABLES**

In case your custom programmed source type should support a generation of extractors, method GENERATE_EXTRACTOR has to be implemented. In addition, a validation should usually be performed before you start generation. To perform this validation method PARAMETER_VALIDATE has to be implemented as well.

**Embedding a specific UI for your source type**

Sometimes you might encounter the situation that the parameter customizing available in Source Type maintenance is not sufficient but that you need to embed your own UI providing further features like source type specific functions you would like to call via buttons. To start the creation of these UIs you first have to develop a ABAP Web Dynpro component implementing Web Dynpro Component Interface "WDCI_DSWP_CDC_PARAMETER". This interface provides the following interface views:

- **W_DSWP_CDC_PARAM_INS**
  This interface view can be embedded into the Comparison Instance screen to maintain and display parameters and source type specific functions.
- **W_DSWP_CDC_PARAM_OBJ**
  This interface view can be embedded into the Comparison Object view to maintain parameters and source type specific functions as needed during object maintenance.
- **W_DSWP_CDC_PARAM_RES**
  This interface view can be embedded into the Comparison Result view to maintain parameters and source type specific functions.

The following interface methods are used for interaction between your UI and the embedding main UI:
- **GET_DATA:**
  The method GET_DATA provides parameters and their values to the main CDC UI via exporting parameter ET_PARAMETER. The importing parameter USE_CASE determines which interface view should provide the needed data respective from where the method is called:
  - O – The method is called for a Comparison Object.
  - I – The method is called for a Comparison Instance.

- **SET_DATA**
  This method is used to provide data from the central UI to the embedded screens, e.g. after loading an Comparison Object. The following parameters contain the data to be displayed:
  - Importing parameter IT_SOURCE_PARAMETER contains a list of parameters and their values that should be displayed.
  - The source system is determined by importing parameter IV_SOURCE_NO. A value of 1 means source 1, a value of 2 means source 2.
  - IV_SOURCE_TYPE contains the technical name of the source type which parameters should be displayed like ABDY or ABAP.
  - The USE_CASE again determines whether data us provided for a Comparison Object or a Comparison Instance.

- **SET_DISPLAY_MODE**
  This interface method will be called from the UI whenever a change of the display mode is triggered by the central CDC UI. It only contains Boolean importing parameter IV_DISPLAY_MODE. A value of true means that the embedded UI should be in read-only mode.

After the ABAP Web Dynpro UI to be embedded has been created it has to be embedded in the central screens. To do this easily a new source type parameter “Web Dynpro Component” has been introduced with SAP Solution Manager 7.1 SP12. Simply enter the name of your component into this field for those source types that should utilize the UI. If the field is empty, the standard UIs for parameter maintenance will be called.
APPENDIX

Enhancing SQL extractions with DISTINCT
Sometimes you are not interested to compare all data between two systems but are only interested whether a data with a specific non-unique key exists at all. As the key for the Comparison Run has to be unique you can add the SQL-command DISTINCT to extract key values only once.

Business Example
You replicate orders from one system to the other system where status handling is done. You are not interested in the status information of orders, but would just like to know that at least one status record of a relevant subset Status1 to Status5 exists. This would lead to a statement of the following form (manually added parts in **bold italic**):

```sql
SELECT DISTINCT TABLE1.ORDER_NUM
FROM /*SCHEMA*/TABLE1
JOIN /*SCHEMA*/TABLE2
ON TABLE2.FELD2 = TABLE1.FELD1
WHERE TABLE2.NAME = 'Status1'
OR TABLE2.NAME = 'Status2'
OR TABLE2.NAME = 'Status3'
OR TABLE2.NAME = 'Status4'
OR TABLE2.NAME = 'Status5'
ORDER BY TABLE1.ORDER_NUM
```

TABLE1 would be the table with the order information, while TABLE2 would contain the status information.

Enhancing SQL extractions with relative date selections
A common business case for data comparison is to compare not the complete set of data contained in a system but to restrict the comparison to data that was created or changed within the last days only.

Business Example
You replicate orders from one system to the other system. You are not interested in outdated orders, but would just like to know whether orders created in the last 7 days are consistent.

As the data functions are database dependent you have to enhance the SQL-statement with functions native to the respective database.

Note: As of ST 7.1 SP5, the relative date filtering is available in standard.

**MS-SQL server**
To perform a relative calculation on dates some build-in functions have to be used:
- **GETDATE**
  This function provides the current date and time
- **DATEADD**
  This function can be used to add (or subtract) days to a given date
- **DATEDIFF** needs to be used as well as GETDATE returns the current time as well but we want to restrict the restriction to complete days.

The template for the restriction to 7 days would then be (added parts in **bold italic**):

```sql
SELECT TABLE1.ORDER_NUM
FROM /*SCHEMA*/table1
WHERE (table1.entry_time >= DATEADD(day,-7,DATEDIFF(Day, 0, GETDATE())))
ORDER BY TABLE1.ORDER_NUM
```
Again table1 would be the table containing the order information, order_num would be the primary key (order number) and entry_time would be the time the order was entered.

**MaxDB**
The template for the restriction to 7 days would then be (added parts in *bold italic*

```sql
SELECT TABLE1.ORDER_NUM
FROM /*SCHEMA*/table1
WHERE (table1.entry_time >= SUBDATE(DATE, 7))
ORDER BY TABLE1.ORDER_NUM
```

*Enhanced ABAP data extractor with relative date selection on yesterday (simple date field)*

Here a local data calculation and usage as filter criteria during data selection was added. Added coding is written in *bold italic*.

**FUNCTION** ZCDC_BUSTRANS_DELIVERY_V2.

```sql
*"Local Interface:
*" IMPORTING
*" VALUE(IV_PREV_KEY) TYPE STRING OPTIONAL
*" VALUE(IV_BLOCK_SIZE) TYPE I DEFAULT 1000
*" VALUE(IT_FILTER) TYPE DDSHSELOPS OPTIONAL
*" EXPORTING
*" VALUE(EV_ACT_KEY) TYPE STRING
*" VALUE(EV_BLOCK) TYPE BOOLEAN
*" VALUE(EV_TOTAL) TYPE I
*" VALUE(EV_LINES) TYPE I
*" VALUE(EV_DATA_STRING) TYPE STRING
*" VALUE(EV_DATA_XSTRING) TYPE XSTRING
*" EXCEPTIONS
*" AUTHORIZATION_ERROR
*" DATABASE_ERROR
*" FILTER_ERROR
*" TRANSFORMATION_ERROR
*" CONVERSION_ERROR
*"---------------------------------------------
*"----- CDC Extract Function -----*
*" This function module has been generated.
*" Generated on 2012/03/22 15:15:26
*" Enhanced on 2012/03/27 for custom-specific extractor logic
*" => Please do not overwrite by new generation
*"---------------------------------------------
*" TYPES:
*" BEGIN OF TS_SOURCE_KEY,
*" KEY1 TYPE VBELN,
*" END OF TS_SOURCE_KEY,
*" BEGIN OF TS_SOURCE_DATA,
*" KEY1 TYPE VBELN,
*" END OF TS_SOURCE_DATA.
*" CONSTANTS:
*" LC_DELIMITER TYPE STRING VALUE '|||'.
*" DATA:
CROSS DATABASE COMPARISON – ENHANCEMENT GUIDE

LT_SOURCE_DATA TYPE STANDARD TABLE OF TS_SOURCE_DATA,
LT_TEMP_SOURCE_DATA TYPE STANDARD TABLE OF TS_SOURCE_DATA,
LV_REMAINING_BLOCK_SIZE TYPE I,
LS_SOURCE_DATA TYPE TS_SOURCE_DATA,
LS_SOURCE_KEY TYPE TS_SOURCE_KEY.

DATA:
LV_TABLE_GROUP TYPE BRGRU,
LV_TABLE_GROUP_NO_AUTH TYPE SYMSGV,
LV_ERROR_MESSAGE TYPE STRING,
LV_FILTER_FIELD TYPE STRING,
LX_ROOT TYPE REF TO CX_ROOT.

DATA:
LV_TYPE TYPE STRING,
LV_NO_COMPONENT TYPE I,
LV_KEY_STRING TYPE STRING,
LV_OFFSET TYPE I,
LV_LENGTH TYPE I.

DATA:
LV_YESTERDAY LIKE LIKP-ERDAT.

FIELD-SYMBOLS:
<LS_FILTER_LINE> TYPE DDSHSELOPT,
<FS_FIELD> TYPE ANY.

*** Part 1: Authorization Checks ***

* Authorization check for transaction code SE16
  AUTHORITY-CHECK OBJECT 'S_TCODE' ID 'TCD' FIELD 'SE16'.
  IF SY-SUBRC NE 0.
  * Dummy message to populate SY-MSG* fields (accessible after RFC)
    MESSAGE E001(00) WITH SY-UNAME 'S_TCODE' 'SE16'
    RAISING AUTHORIZATION_ERROR.           "#EC MG_PAR_CNT
  ENDIF.

* Authorization checks for table authorization classes of source tables

SELECT SINGLE CCLASS
FROM TDDAT
INTO LV_TABLE_GROUP
WHERE TABNAME = 'VBUK'.

IF SY-SUBRC NE 0 OR LV_TABLE_GROUP IS INITIAL.
  LV_TABLE_GROUP = '#NC&'.
ENDIF.

  AUTHORITY-CHECK OBJECT 'S_TABU_DIS'
  ID 'DICBERCLS' FIELD LV_TABLE_GROUP
  ID 'ACTVT' FIELD '03'.

  IF SY-SUBRC NE 0.
    CONCATENATE LV_TABLE_GROUP_NO_AUTH LV_TABLE_GROUP
    INTO LV_TABLE_GROUP_NO_AUTH SEPARATED BY ','.
  ENDIF.

  SELECT SINGLE CCLASS
  FROM TDDAT
INTO LV_TABLE_GROUP
WHERE TABNAME = 'LIKp'.

IF SY-SUBRC NE 0 OR LV_TABLE_GROUP IS INITIAL.
   LV_TABLE_GROUP = '&NC&'.
ENDIF.

AUTHORITY-CHECK OBJECT 'S_TABU_DIS'
   ID 'DICBERCLS' FIELD LV_TABLE_GROUP
   ID 'ACTVT' FIELD '03'.

IF SY-SUBRC NE 0.
   CONCATENATE LV_TABLE_GROUP_NO_AUTH LV_TABLE_GROUP
      INTO LV_TABLE_GROUP_NO_AUTH SEPARATED BY ','.
ENDIF.

* Common exception with all missing table groups
IF LV_TABLE_GROUP_NO_AUTH IS NOT INITIAL.
   SHIFT LV_TABLE_GROUP_NO_AUTH LEFT.
* Dummy message to populate SY-MSG* fields (accessible after RFC)
   MESSAGE E001(00) WITH SY-UNAME 'S_TABU_DIS' LV_TABLE_GROUP_NO_AUTH
      RAISING AUTHORIZATION_ERROR. "#EC MG_PAR_CNT"
ENDIF.

*** Part 2: Creation of Filter Range Tables ***

* Instead of variable CDC filters, we have a hard-coded
* filter on yesterday's date!
   LV_YESTERDAY = SY-DATUM - 1.

*** Part 3: Source Data Extraction ***

IF IV_BLOCK_SIZE < 0.
* Count expected number of rows only
   SELECT COUNT(*)
      INTO EV_TOTAL
      FROM VBUK JOIN LIKP ON LIKP~VBELN = VBUK~VBELN
      WHERE VBUK~WBSTK EQ 'C'
      AND LIKP~ERDAT = LV_YESTERDAY
      AND LIKP~VKORG = '0001'.
   RETURN.
ENDIF.

* Extract source data
* Move imported key string to key structure
   DESCRIBE FIELD LS_SOURCE_KEY
      TYPE LV_TYPE
      COMPONENTS LV_NO_COMPONENT.
IF IV_PREV_KEY IS NOT INITIAL.
   TRY.
      DO LV_NO_COMPONENT TIMES.
         ASSIGN COMPONENT SY-INDEX
            OF STRUCTURE LS_SOURCE_KEY
            TO <FS_FIELD>.
         IF SY-INDEX EQ 1.
            LV_KEY_STRING = IV_PREV_KEY.
         ELSE.
            LV_KEY_STRING = LV_KEY_STRING+LV_OFFSET.
            LV_KEY_STRING = LV_KEY_STRING+LV_LENGTH.
ENDIF.

FIND LC_DELIMITER
  IN LV_KEY_STRING
  MATCH OFFSET LV_OFFSET
  MATCH LENGTH LV_LENGTH.
ENDIF.

CATCH CX_SY_CONVERSION_ERROR INTO LX_ROOT.
MESSAGE E001(00) WITH IV_PREV_KEY
  RAISING CONVERSION_ERROR.
  "#EC MG_PAR_CNT

ENDTRY.
ENDDO.

SELECT VBUK~VBELN
  INTO TABLE LT_SOURCE_DATA
  UP TO IV_BLOCK_SIZE ROWS
  FROM VBUK JOIN LIKP ON LIKP~VBELN = VBUK~VBELN
  WHERE VBUK~WBSTK = 'C'
    AND LIKP~ERDAT = LV_YESTERDAY
    AND LIKP~VKORG = '0001'
    AND VBUK~VBELN > LS_SOURCE_KEY-KEY1
    ORDER BY VBUK~VBELN.

EV_LINES = SY_DBCNT.

* Indicate whether more blocks are to be expected
IF EV_LINES < IV_BLOCK_SIZE OR IV_BLOCK_SIZE = 0.
  CLEAR EV_BLOCK.
ELSE.
  EV_BLOCK = 'X'.
ENDIF.

* Get key of last source data row
READ TABLE LT_SOURCE_DATA INDEX EV_LINES INTO LS_SOURCE_DATA.
MOVE-CORRESPONDING LS_SOURCE_DATA TO LS_SOURCE_KEY.

* Move key structure in export key string
DO LV_NO_COMPONENT TIMES.
  ASSIGN COMPONENT SY_INDEX OF STRUCTURE LS_SOURCE_KEY TO <FS_FIELD>.
  IF SY_INDEX EQ 1.
    EV_ACT_KEY = <FS_FIELD>.
  ELSE.
    CONCATENATE EV_ACT_KEY <FS_FIELD>
      INTO EV_ACT_KEY
      SEPARATED BY LC_DELIMITER.
  ENDIF.
ENDDO.

*** Part 4: XML Transformation ***
TRY.
* Call transformation from ABAP internal table to XML string (char)
  IF EV_DATA_STRING IS REQUESTED.
    CALL TRANSFORMATION ID
**Source Table** = LT_SOURCE_DATA

RESULT XML EV_DATA_STRING.

* Call transformation from ABAP internal table to XML Xstring (binary)
ELSEIF EV_DATA_XSTRING IS REQUESTED.
    CALL TRANSFORMATION ID
    SOURCE TABLE = LT_SOURCE_DATA
    RESULT XML EV_DATA_XSTRING.
ENDIF.

* Catch cx_root as cx_transformation_error is only available as of bas
  is version 7.0
    CATCH CX_ROOT INTO LX_ROOT.
    LV_ERROR_MESSAGE = LX_ROOT->GET_TEXT( ).
* Dummy message to populate SY-MSG* fields (accessible after RFC)
    MESSAGE E001(00) WITH LV_ERROR_MESSAGE
        RAISING TRANSFORMATION_ERROR.  "#EC MG_PAR_CNT
ENDTRY.

**Endfunction.**

**Enhanced ABAP data extractor with relative date selection on yesterday (timestamp field), independent SELECT... FOR ALL ENTRIES, additional application logic, key mapping**

**Function** ZCDC_BUSTRANS_DELIVERY_V2.

*------------------------------------------------------------------------------------------------------------------
**Local Interface:**
* IMPORTING
  ** VALUE(IV_PREV_KEY) TYPE STRING OPTIONAL
  ** VALUE(IV_BLOCK_SIZE) TYPE I DEFAULT 1000
  ** VALUE(IT_FILTER) TYPE DDSHSELOPS OPTIONAL
* EXPORTING
  ** VALUE(EV_ACT_KEY) TYPE STRING
  ** VALUE(EV_BLOCK) TYPE BOOLEAN
  ** VALUE(EV_TOTAL) TYPE I
  ** VALUE(EV_LINES) TYPE I
  ** VALUE(EV_DATA_STRING) TYPE STRING
  ** VALUE(EV_DATA_XSTRING) TYPE XSTRING
* EXCEPTIONS
  ** AUTHORIZATION_ERROR
  ** DATABASE_ERROR
  ** FILTER_ERROR
  ** TRANSFORMATION_ERROR
  ** CONVERSION_ERROR
*------------------------------------------------------------------------------------------------------------------

---- CDC Extract Function ----
* This function module has been generated.
* Generated on 2012/03/22 15:13:44
* Enhanced on 2012/03/27 for custom-specific extractor logic
* => Please do not overwrite by new generation

**Types:**

BEGIN OF ts_source_key,
    KEY1 TYPE /SCDL/DL_REFDOCNO,
END OF ts_source_key,

BEGIN OF ts_source_data,
    KEY1 TYPE /SCDL/DL_REFDOCNO,
* DATA1 TYPE /SCDL/DL_CRETST,
  * DATA2 TYPE /SCDL/DL_STATUS_VALUE,
  END OF ts_source_data,
  BEGIN OF ts_del_item_status,
    DOCID  TYPE /SCDL/DL_DOCID,
    ITEMID TYPE /SCDL/DL_ITEMID,
    STATUS_VALUE  TYPE /SCDL/DL_STATUS_VALUE,
  END OF ts_del_item_status,
  tt_del_item_status TYPE SORTED TABLE OF ts_del_item_status
  WITH UNIQUE KEY docid itemid.
  CONSTANTS:
    lc_gi_finished  TYPE /SCDL/DL_STATUS_VALUE VALUE '9'.
  CONSTANTS:
    lc_delimiter  TYPE string VALUE '|||'.
  DATA:
    lt_source_data  TYPE STANDARD TABLE OF ts_source_data,
    lt_temp_source_data  TYPE STANDARD TABLE OF ts_source_data,
    lv_remaining_block_size  TYPE i,
    ls_source_data  TYPE ts_source_data,
    ls_source_key  TYPE ts_source_key.
  DATA:
    lv_table_group  TYPE brgru,
    lv_table_group_no_auth  TYPE symsgv,
    lv_error_message  TYPE string,
    lv_filter_field  TYPE string,
    lx_root  TYPE REF TO cx_root.
  * DATA:
  * lt_FILTER1 TYPE RANGE OF /SCDL/DL_CRETST, ls_FILTER1 LIKE LINE OF *lt_FILTER1.
  DATA:
    lv_type  TYPE string,
    lv_no_component  TYPE i,
    lv_key_string  TYPE string,
    lv_offset  TYPE i,
    lv_length  TYPE i.
  DATA:
    lv_tstamp_from  LIKE /scdl/db_proch_o-cretst,
    lv_tstamp_to  LIKE /scdl/db_proch_o-cretst,
    lv_timezone  LIKE ttzz-tzone VALUE 'UTC',
    lv_time  TYPE sy-zeit,
    lv_yesterday  TYPE sy-datum,
  it_del_doc  TYPE STANDARD TABLE OF /scdl/db_proch_o-docid,
  it_del_item_status  TYPE tt_del_item_status,
  it_del_gi_finished  TYPE STANDARD TABLE OF /scdl/dl_docid,
  lv_status_low  TYPE /scdl/dl_status_value.
FIELD-SYMBOLS:
  <ls_filter_line>  TYPE ddshselopt,
  <fs_field>  TYPE any,
  <ls_del_item_status>  TYPE ts_del_item_status.

*** Part 1: Authorization Checks ***
* Authorization check for transaction code SE16
AUTHORITY-CHECK OBJECT 'S_TCODE' ID 'TCD' FIELD 'SE16'.
IF sy-subrc NE 0.
* Dummy message to populate SY-MSG** fields (accessible after RFC)
MESSAGE e001(00) WITH sy-uname 'S_TCODE' 'SE16'
  RAISING authorization_error.
"#EC MG_PAR_CNT
ENDIF.

* Authorization checks for table authorization classes of source tables

SELECT SINGLE cclass
FROM tddat
INTO lv_table_group
WHERE tabname = '/SCDL/DB_REFDOC'.
IF sy-subrc NE 0 OR lv_table_group IS INITIAL.
  lv_table_group = '&NC&'.
ENDIF.

AUTHORITY-CHECK OBJECT 'S_TABU_DIS'
  ID 'DICBERCLS' FIELD lv_table_group
  ID 'ACTVT' FIELD '03'.

IF sy-subrc NE 0.
  CONCATENATE lv_table_group_no_auth lv_table_group
    INTO lv_table_group_no_auth SEPARATED BY ','.
ENDIF.

SELECT SINGLE cclass
FROM tddat
INTO lv_table_group
WHERE tabname = '/SCDL/DB_PROCH_O'.
IF sy-subrc NE 0 OR lv_table_group IS INITIAL.
  lv_table_group = '&NC&'.
ENDIF.

AUTHORITY-CHECK OBJECT 'S_TABU_DIS'
  ID 'DICBERCLS' FIELD lv_table_group
  ID 'ACTVT' FIELD '03'.

IF sy-subrc NE 0.
  CONCATENATE lv_table_group_no_auth lv_table_group
    INTO lv_table_group_no_auth SEPARATED BY ','.
ENDIF.

SELECT SINGLE cclass
FROM tddat
INTO lv_table_group
WHERE tabname = '/SCDL/DB_STATUS'.
IF sy-subrc NE 0 OR lv_table_group IS INITIAL.
  lv_table_group = '&NC&'.
ENDIF.

AUTHORITY-CHECK OBJECT 'S_TABU_DIS'
  ID 'DICBERCLS' FIELD lv_table_group
  ID 'ACTVT' FIELD '03'.

IF sy-subrc NE 0.
   CONCATENATE lv_table_group_no_auth lv_table_group
       INTO lv_table_group_no_auth SEPARATED BY ',', '.
ENDIF.

* Common exception with all missing table groups
IF lv_table_group_no_auth IS NOT INITIAL.
   SHIFT lv_table_group_no_auth LEFT.
* Dummy message to populate SY-MSG* fields (accessible after RFC)
   MESSAGE E001(00) WITH sy-uname 'S_TABU_DIS' lv_table_group_no_auth
      RAISING authorization_error. "#EC MG_PAR_CNT
ENDIF.

*** Part 2: Creation of Filter Range Tables ***
* Copy generic select-options into type-conform range tables
* LOOP AT it_filter ASSIGNING <ls_filter_line>.
* IF <ls_filter_line>-shlpname = '/SCDL/DB_PROCH_O' AND
*   <ls_filter_line>-shlpfield = 'CRETST'. MOVE-CORRESPONDING
*   <ls_filter_line> TO ls_FILTER1. APPEND ls_FILTER1 TO lt_FILTER1.
* ELSE.
*   CONCATENATE <ls_filter_line>-shlpname <ls_filter_line>-shlpfield
*       INTO lv_filter_field SEPARATED BY ', '.
*   *** Dummy message to populate SY-MSG* fields (accessible after RFC)
*   MESSAGE E001(00) WITH lv_filter_field
*       RAISING filter_error. "#EC MG_PAR_CNT
* ENDIF.
* ENDDO.

*** Part 3: Source Data Extraction ***
IF iv_block_size < 0.
** Count expected number of rows only
* SELECT COUNT(*)
* INTO ev_total
* FROM /SCDL/DB_REFDOC
* JOIN /SCDL/DB_PROCH_O
* ON /SCDL/DB_PROCH_O-DOCID = /SCDL/DB_REFDOC-DOCID
* JOIN /SCDL/DB_STATUS
* ON /SCDL/DB_STATUS-DOCID = /SCDL/DB_PROCH_O-DOCID
* WHERE /SCDL/DB_REFDOC-REFDOCCAT EQ 'ERP'
* AND /SCDL/DB_PROCH_O-DOCCAT EQ 'PDO'
* AND /SCDL/DB_STATUS-ITEMID EQ '00000000000000000000000000000000'
* AND /SCDL/DB_STATUS-STATUS_TYPE EQ 'DGI'
* AND /SCDL/DB_PROCH_O-CRETST BETWEEN lv_tstamp_from
*      AND lv_tstamp_to.

Instead of variable CDC filters, we have a hard-coded
* Filter on yesterday's date:
lv_yesterday = sy-datum - 1
CLEAR lv_time.
CONVERT DATE lv_yesterday
   TIME lv_time
   INTO TIME STAMP lv_tstamp_from TIME_ZONE lv_timezone.
lv_time = '235959'.
CONVERT DATE lv_yesterday
   TIME lv_time
   INTO TIME STAMP lv_tstamp_to TIME_ZONE lv_timezone.
Counting would required same complex logic as extract itself => too expensive, just return 0.

RETURN.

Extract source data

Cannot use multiple block fetches with key offset (different keys)
* => need to fetch all deliveries in one block
* Move imported key string to key structure
* DESCRIPT FIELD ls_source_key
* TYPE lv_type
* COMPONENTS lv_no_component.
* IF lv_prev_key IS NOT INITIAL.
  * TRY:
  * DO lv_no_component TIMES
  * ASSIGN COMPONENT sy-index
    OF STRUCTURE ls_source_key
    TO <fs_field>.
  * IF sy-index EQ 1.
    * lv_key_string = lv_prev_key.
  * ELSE.
    * lv_key_string = lv_key_string+lv_offset.
    * lv_key_string = lv_key_string+lv_length.
  * ENDIF.
  * FIND lcDelimiter
    IN lv_key_string
    MATCH OFFSET lv_offset
    MATCH LENGTH lv_length.
  * IF sy-subrc EQ 0.
    * MOVE lv_key_string(lv_offset) TO <fs_field>.
  * ELSE.
    * MOVE lv_key_string TO <fs_field>.
    * EXIT.
    * ENDF.
  * ENDDO.
  * CATCH cx_sy_conversion_error INTO lx_root.
    * MESSAGE e001(00) WITH lv_prev_key
      RAISING conversion_error. "#EC MG_PAR_CNT"
  * ENTRY.
  * ENDF.

This was the original generated join, but cannot be used here
* SELECT /SCDL/DB_REFDOC-REFDOCNO
  INTO TABLE lt_source_data
  UP TO lv_block_size ROWS
  FROM /SCDL/DB_REFDOC
  JOIN /SCDL/DB_PROCH_O
    ON /SCDL/DB_PROCH_O-DOCID = /SCDL/DB_REFDOC-DOCID
  JOIN /SCDL/DB_STATUS
    ON /SCDL/DB_STATUS-DOCID = /SCDL/DB_PROCH_O-DOCID
  WHERE /SCDL/DB_REFDOC-REFDOCCAT EQ 'ERP'
    AND /SCDL/DB_PROCH_O-DOCCAT EQ 'PDO'
    AND /SCDL/DB_STATUS-ITEMID EQ '00000000000000000000000000000000'
    AND /SCDL/DB_STATUS-STATUS_TYPE EQ 'DGI'
    AND /SCDL/DB_PROCH_O-CRETST BETWEEN lv_tstamp_from AND lv_tstamp_to
    AND /SCDL/DB_REFDOC-REFDOCNO > ls_source_key-key1
  ORDER BY /SCDL/DB_REFDOC-REFDOCNO.
CROSS DATABASE COMPARISON – ENHANCEMENT GUIDE

* Customer Extractor Logic (with aggregation of item status)

```
SELECT DOCID
FROM /SCDL/DB_PROCH_O
INTO TABLE lt_del_doc
WHERE DOCCAT = 'PDO'
  AND CRETST BETWEEN lv_tstamp_from AND lv_tstamp_to
ORDER BY DOCID.
```

* Get all outbound delivery headers created yesterday

```
SELECT DOCID
FROM /SCDL/DB_PROCH_O
INTO TABLE lt_del_doc
WHERE DOCCAT = 'PDO'
  AND CRETST BETWEEN lv_tstamp_from AND lv_tstamp_to
ORDER BY DOCID.
```

* Get all items for these delivery headers

```
IF lt_del_doc IS NOT INITIAL.
  SELECT DOCID ITEMID STATUS_VALUE
    FROM /SCDL/DB_STATUS
    INTO CORRESPONDING FIELDS OF TABLE lt_del_item_status
    FOR ALL ENTRIES IN lt_del_doc
    WHERE DOCID = lt_del_doc-table_line
      AND ITEMID <> '00000000000000000000000000000000'
      AND STATUS_TYPE = 'DGI'.
ENDIF.
```

* Find all deliveries where all items have goods issue completed

```
lv_status_low = lc_gi_finished.
LOOP AT lt_del_item_status ASSIGNING <ls_del_item_status>.
  Find any item status smaller than 9 (goods issue finished)
  IF <ls_del_item_status>-status_value < lv_status_low,
    lv_status_low = <ls_del_item_status>-status_value.
  ENDIF.

  Remember all deliveries with all items completed
  AT END OF docid.
  IF lv_status_low = lc_gi_finished.
    APPEND <ls_del_item_status>-docid TO lt_del_gi_finished.
  ENDIF.
  lv_status_low = lc_gi_finished.
ENDAT.
ENDLOOP.
```

You do not document whether the ERP delivery numbers are:

```
IF lt_del_gi_fin IS NOT INITIAL.
  SELECT REFDOCNO
    FROM /SCDL/DB_REFDOC
    INTO TABLE lt_source_data
    FOR ALL ENTRIES IN lt_del_gi_finished
    WHERE DOCID = lt_del_gi_finished-table_line
      AND REFDOCCAT = ' ERP'.
  ev_lines = sy-dbcnt.
  SORT lt_source_data.
ENDIF.
```

** Indicate whether more blocks are to be expected

```
* IF ev_lines < lv_block_size OR lv_block_size = 0.
  CLEAR ev_block.
  ELSE.
  ev_block = 'X'.
  ENDIF.
```

* Cannot use multiple block fetches with key offset. Fetch all blocks
  together to process all deliveries in one block.
CROSS DATABASE COMPARISON – ENHANCEMENT GUIDE

CLEAN ev_block.

* Get key of last source data row
READ TABLE lt_source_data INDEX ev_lines INTO ls_source_data.
MOVE-CORRESPONDING ls_source_data TO ls_source_key.

* Move key structure in export key string
DO lv_no_component TIMES.
   ASSIGN COMPONENT sy-index OF STRUCTURE ls_source_key TO <fs_field>.
   IF sy-index EQ 1.
      ev_act_key = <fs_field>.
   ELSE.
      CONCATENATE ev_act_key <fs_field>
      INTO ev_act_key
      SEPARATED BY lc_delimiter.
   ENDIF.
ENDDO.

*** Part 4: XML Transformation ***
TRY.
* Call transformation from ABAP internal table to XML string (char)
   IF ev_data_string IS REQUESTED.
   CALL TRANSFORMATION id
   SOURCE table = lt_source_data
   RESULT XML ev_data_string.

* Call transformation from ABAP internal table to XML Xstring (binary)
   ELSEIF ev_data_xstring IS REQUESTED.
   CALL TRANSFORMATION id
   SOURCE table = lt_source_data
   RESULT XML ev_data_xstring.
ENDIF.

* Catch cx_root as cx_transformation_error is only available as of bas
   *is version 7.0
   CATCH cx_root INTO lx_root.
   lv_error_message = lx_root->get_text().
   * Dummy message to populate SY-MSG* fields (accessible after RFC)
   MESSAGE e001(00) WITH lv_error_message
   RAISING transformation_error. "#EC MG_PAR_CNT
ENDTRY.

ENDFUNCTION.
**BAdI implementation for enhancement spot DSWP_CDC_SOURCE**

**Business Use Case**
Cross database compare is setup to diagnose the data inconsistencies between two source systems, potentially between SAP – SAP and SAP – Non-SAP. During the data compare there are comparison fields that have different formats, example: one comparison field which has leading zeroes and corresponding comparison field in other source has no leading zeroes. In this case, the format for the field in one system needs to be converted so both source 1 and source2 have same format and comparison runs successfully. Solution for this scenario is to extend the Cross Database Compare tool by implementing the enhancement spot provided by the standard: DSWP_CDC_SOURCE

**Procedure**
Follow the steps to implement the CDC Extension using enhancement spot implementation
- Go to transaction SE18 (BAdI Builder) and select the enhancement spot for implementation → DSWP_CDC_SOURCE and Display
- Select BAdI Definition and right click to choose Create BAdI Implementation as shown in the screen

![Figure 6](image)

- In the next dialog choose Create Enhancement Implementation(F8). Each BAdI implementation has to be assigned to an enhancement implementation.

![Figure 7](image)

- Input technical name, short text for enhancement implementation and hit enter
- Maintain the Object attributes – Local object if this implementation is not expected to be transported.

- Please note that ZDSWP_CDC_SOURCE_2 implementation is just created

- Create Implementation Class: ZCL_DSWP_CDC_SOURCE2 for the BAdI Implementation
• Choose Copy Sample Class

![Figure 12](image1.png)

Figure 12

• Review the properties for the enhancement implementation that was just created. Status is Inactive

![Figure 13](image2.png)

Figure 13
- Navigate to the Class Interface you have created Z_CL_DSWP_CDC_SOURCE2

![Figure 14](image1.png)

- Provide the implementation on Method: MODIFY_DATA_TABLE of Object Type: IF_DSWP_CDC_SOURCE_DATA

![Figure 15](image2.png)

```
method IF_DSWP_CDC_SOURCE_DATA-MODIFY_DATA_TABLE.

* In the following example, the comparison object has in column 3 a data type NUMC. So the content value has leading zeros, which need to be converted to an integer number for comparison reasons.

DATA:
    lv_integer TYPE i,
    lv_number TYPE n,
    lv_txt TYPE c LENGTH 7.

FIELD-SYMBOLS:
    <ls_source_row> TYPE any, "row of source data table"
    <lv_source_val> TYPE any, "component of table row"

*Example: Extension Code Begin..!!

* Loop over each line of the source system data block
LOOP AT ct_source_tab ASSIGNING <ls_source_row>.
  * Access content of column number 1
  ASSIGN COMPONENT 1 OF STRUCTURE <ls_source_row> TO <lv_source_val>.
  IF <lv_source_val> IS ASSIGNED.
  * Cast string value with leading zeros into a real integer type
    TRY.
      lv_integer = <lv_source_val>.
```

SAP
lv_number = <lv_source_val>.
lv_txt = <lv_source_val>.

<lv_source_val> = lv_integer.

* If it cannot be converted into a valid number, clear the value
CATCH cx_sy_conversion_no_number.
  CLEAR <lv_source_val>.
  ENTR.
  ENDF.
  ENDL.

* Example: Extension Code End.!!

endmethod.

- Save and Maintain the BAdi Implementation Filter Values specific to the Comparison Object or Comparison Instance, and relevant Source 1 or Source2

![Figure 16](image)

- Save, and Activate all the objects part of the Enhancement Implementation

![Figure 17](image)

- Execute the comparison run for comparison instance. Screenshot below shows the Number format conversion (debug mode)
Figure 18
• Comparison Results before implementation of CDC specific BAdI : Object Key: VBELN_OBJECTID has leading zeroes

![Comparison Results](image)

**Figure 19**

• Result after BAdI implementation – Converted to Integer

After the conversion, Object Key: VBELN_OBJECTID has no leading zeroes
### Global Result Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison Instance</td>
<td>Z_TEST7_SALESORDERS</td>
</tr>
<tr>
<td>Comparison Instance Desc.</td>
<td>SALES ORDER LIMITERS</td>
</tr>
<tr>
<td>Start Date</td>
<td>06/12/2012 22:08:30</td>
</tr>
<tr>
<td>End Date</td>
<td>06/12/2012 22:08:37</td>
</tr>
<tr>
<td>System 1 Description</td>
<td>SIS_MBOCLNTT7D0_READ</td>
</tr>
<tr>
<td>System 2 Description</td>
<td>SIS_CPOCLNTT100_READ</td>
</tr>
<tr>
<td>Run ID</td>
<td>0030080907</td>
</tr>
<tr>
<td>Run Time</td>
<td>10:241</td>
</tr>
</tbody>
</table>

### Result Overview

<table>
<thead>
<tr>
<th>Type of Inconsistency</th>
<th>Number of Objects</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects Compared</td>
<td>18,241</td>
<td>100.00%</td>
</tr>
<tr>
<td>Objects Identical</td>
<td>0</td>
<td>0.00%</td>
</tr>
<tr>
<td>Objects existing in System 1 only</td>
<td>27</td>
<td>0.15%</td>
</tr>
<tr>
<td>Objects existing in System 2 only</td>
<td>18,214</td>
<td>99.85%</td>
</tr>
<tr>
<td>Objects with differences</td>
<td>0</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

**Message Text:** The comparison instance run finished successfully. Progress: 100.00%

### Result Details

<table>
<thead>
<tr>
<th>Comparison Object Key (VISUAL_OBJECT)</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>300002</td>
<td></td>
</tr>
<tr>
<td>300001</td>
<td></td>
</tr>
<tr>
<td>300011</td>
<td></td>
</tr>
<tr>
<td>300012</td>
<td></td>
</tr>
<tr>
<td>300013</td>
<td></td>
</tr>
<tr>
<td>300014</td>
<td></td>
</tr>
<tr>
<td>200015</td>
<td></td>
</tr>
<tr>
<td>200016</td>
<td></td>
</tr>
<tr>
<td>200017</td>
<td></td>
</tr>
<tr>
<td>200018</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 20**