**Cookbook**
SAP S/4HANA
Document Version: 1.2

**CUSTOMER**

# Required Code and DDIC Changes for Customers using Pricing and Condition Technique in SAP S/4HANA

## Cookbook for Customer Adoption

# Document History

| Version | Date | Change |
|---|---|---|
| 1.0 | 2015-10-15 | Final |
| 1.1 | 2015-12-18 | Chapter 2.1: Table of changed data elements has been enhanced by data elements KALKS and KALVG; Chapter 2.3.1: Further information related to BAPIs, IDOCs, RFC enabled function modules; Chapter 3.1 has been enhanced by additional changes |
| 1.2 | 2016-04-27 | Chapter 2.2: Customer own fields in KONV to be appended to structure PRCS_ELEMENTS_DATA not table PRCD_ELEMENTS |

# Table of Contents

# 1 Introduction

In SAP S/4HANA, the data model of pricing and of the condition technique has been changed. In some cases, the change requires the adaptation of the customer code and dictionary, or at least checking whether code or the dictionary need to be adapted.  This cookbook aims to help users to understand the changes and their consequences.

# 2 Changes in Pricing

Business documents within the SAP Business Suite, such as the sales order or the purchase order, are used to store the pricing result in database table KONV. For SAP S/4HANA, the definition and properties of the fields in table KONV were reviewed. Based on customer feedback and SAP experience, this review led to the following results:

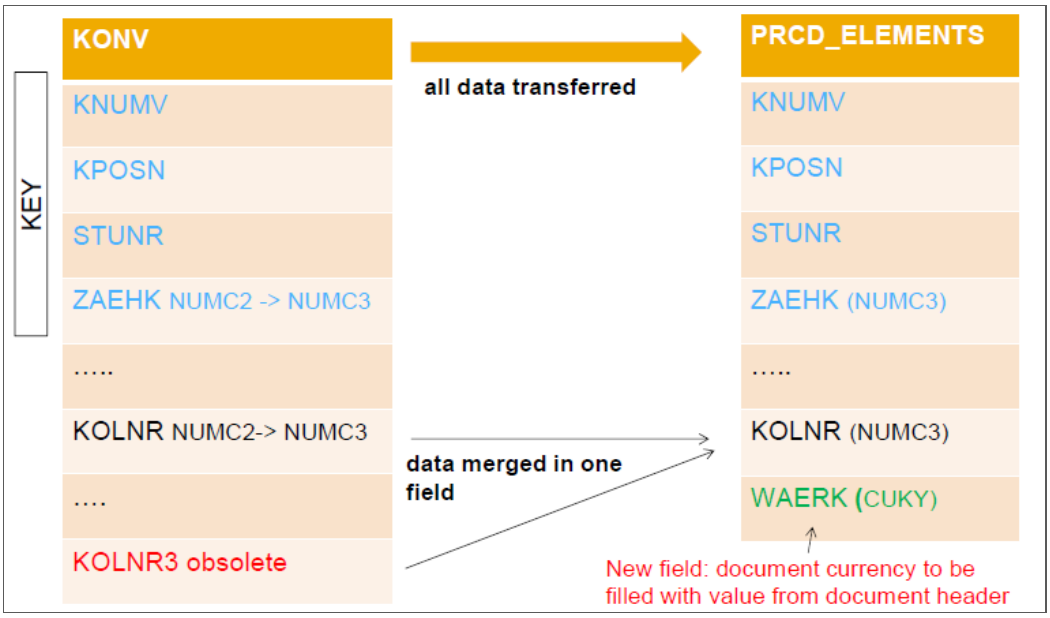
- Several fields have been extended, but this is only the case in the database table for document conditions that are to be modified to the required precision level. Note that a different precision level is not yet available and that SAP is not committed to making such a function available in future releases.
- Some fields that are functionally obsolete have been removed from the database table for document conditions.
- Some existing approaches to overcoming length limitations are now part of the standard delivery of the database table for document conditions.

One of SAP's primary design goals was to keep business applications that use pricing stable. Consequently, the technical realization of the above changes has led to the following results:

- A new database table called PRCD_ELEMENTS has been introduced to replace the KONV table's function of storing document conditions.
- The database layer is now clearly separated. Access to it has to be channeled through newly provided APIs and a CDS view.
- All layers above the database layer are still using the classical KONV field properties to ensure compatibility and stability. Therefore, KONV is continued in all these layers as a structure definition in the data dictionary.

## 2.1 Changes in the Data Model

The following figure shows the main differences in the data model of the pricing result data:




The length of several data elements has been changed:

| Data Elements | Description | Former Length | New Length | Remark |
|---|---|---|---|---|
| KOLNR, EVSNR, FSELNR | Number of Access in Access sequence | NUMC2 | NUMC3 | Active |
| DZAEHK, DZAEHK_IND, DZAEKO, BBP_ZAEHK | Condition Counter in Pricing Result | NUMC2 | NUMC3 | Length is still restricted in Coding to NUMC2 to allow exchange with SAP Business Suite, This might change in the future. |
| KOBED, KOFRM, KOFRA, KOFRS | Pricing Formula & Requirement Number | NUMC3 | NUMC7 | Restriction to NUMC3 in VOFM routines still in place. |
| GRLNR | Group condition routine | NUMC2 | NUMC7 | Restriction to NUMC2 still in place. |
| KALKS, KALVG | Customer and document classification for pricing procedure determination | CHAR1 | CHAR2 | Active |

In addition, the data elements of some fields in the new database table PRCD_ELEMENTS have been changed compared to their KONV equivalents.

| Field | Description | Length in KONV | Length in PRCD_ELEMENTS |
|---|---|---|---|
| KRECH | Calculation type for conditions | CHAR 1 | CHAR 3 |
| KAWRT | Condition Base Value | CURR 15,2 | DEC 24,9 |
| KBETR | Condition Rate | CURR 11,2 | DEC 24,9 |
| KUMZA | Numerator for Converting to Base UoM | DEC 5 | DEC 10 |
| KUMNE | Denominator for Converting to Base UoM | DEC 5 | DEC 10 |
| KOPOS | Sequential number of the condition | NUMC 2 | NUMC 3 |
| KWERT | Condition Value | CURR 13,2 | CURR 15,2 |
| KSTBS | Scale Base Value | CURR 15,2 | DEC 24,9 |
| KZBZG | Scale Basis Indicator | CHAR 1 | CHAR 3 |
| KWERT_K | Condition Value | CURR 13,2 | CURR 15,2 |
| KAWRT_K | Condition Base Value | CURR 15,2 | DEC 24,9 |
| KDATU | Timestamp for Pricing Date | DATS 8 | CHAR 14 |

The fields WEGXX and STUFE are no longer supported, and therefore do not exist in PRCD_ELEMENTS.

If you have been using field KOLN3 to overcome length limitations for KOLNR:
Field KOLN3 is no longer needed in PRCD_ELEMENTS, because KOLNR is now of length NUMC3. The content of KONV-KOLNR3 is merged with the content of the extended KOLNR field in PRCD_ELEMENTS during migration.

The pricing result now contains the document currency of the corresponding document (field WAERK). This is because in PRCD_ELEMENTS, some amounts (such as the condition rate or base value) are no longer stored as CURR, but as DEC fields, and hence in their natural form, instead.

Example: A condition rate of 100 JPY (a currency having no decimals) used to be stored in KONV as 1.00, a condition rate of 100 USD (a currency having two decimals) as 100.00. In PRCD_ELEMENTS both will be stored as 100.000000000. A conversion between PRCD_ELEMENTS and the internal format therefore requires the knowledge of all currencies involved.

## 2.2 Impact on Customer DDIC enhancements

If you have added an append structure to database table KONV in the source release, you must also add an append structure with the same fields (same field names, same data types) to the new structure PRCS_ELEMENTS_DATA which is included in database tables PRCD_ELEMENTS and PRCD_ELEM_DRAFT and structure PRCS_ELEMENTS. Do not append your fields to PRCD_ELEMENTS directly. You have to add the append structure to PRCS_ELEMENTS_DATA in conversion phase SPDD.
This is necessary because only then will the automatic data conversion from KONV to PRCD_ELEMENTS convert the content of the append fields to the new database table PRCD_ELEMENTS.
The pre-check delivered via SAP Note 2188735 provides a check for such append fields, and issues a warning if these append fields are recognized in the source release. The warning refers to the necessary action in conversion phase SPDD.

> **Note**
>
> In S/4HANA, tables PRCD_ELEM_DRAFT and MMQTNPRICING_D are used to temporarily store pricing results as a draft during the creation or modification of documents prior to saving or 'activating' these documents.

When reading from PRCD_ELEMENTS into the internal KONV format, in many applications the CDS view V_KONV is used. This view reads data from database table PRCD_ELEMENTS and exposes them in the format of database table KONV. It is therefore necessary to enhance this CDS view, too. Otherwise, customers' own fields will not be automatically transferred when selecting pricing result records. The same is true for table PRCD_ELEM_DRAFT and CDS view V_KONV_DRAFT.

> **Caution**
>
> Customers' -own fields that are assigned to table KONV need to be added to structure PRCS_ELEMENTS_DATA which is included in database table PRCD_ELEMENTS and PRCD_ELEM_DRAFT and structure PRCS_ELEMENTS in the SAP S/4HANA code line. In addition the fields need to be appended to table MMQTNPRICING_D.
>
> By extending the CDS view V_KONV and V_KONV_DRAFT the fields are considered in the CDS views and hence are read in automatically when the pricing result is accessed by the applications. Therefore for customers with own fields in KONV, there is also the need to extend the CDS views.

SAP provides centrally reports to migrate data from the old database table KONV to the new database table PRCD_ELEMENTS. The content of customer specific fields is automatically migrated if the fields exist in the source and in the target table.

## 2.3 Impact on Customer Coding

This chapter describes how to enable customers' own code and integrate it with pricing for SAP S/4HANA. Every possible access type to database table KONV must be reviewed and adjusted. It is essential to replace all INSERT, UPDATE, MODIFY, and DELETE statements for database table KONV.

> **Caution**
>
> It is strictly forbidden to insert, update, or delete any entries in database table KONV directly. You need to use the new pricing result API to access the database layer!

> **Note**
>
> You need to provide the document currency for writing to the new database table PRCD_ELEMENTS using the new pricing result API. For more information, see the API documentation and code examples sections.

A select on the database table KONV is also no longer possible since it would not contain any data.

You should use the provided API. To access the price result within custom CDS views or in more complex select statements not covered by the API interface, you can use the CDS view named V_KONV instead.

> **Note**
>
> SAP advises you not to use the CDS view for simple selections of the pricing results. PRCD_ELEMENTS was created as a transparent table, therefore automatic buffering for cluster tables is no longer available. SAP recommends that you use the provided API.

As mentioned above, the new persistency needs the document currency of the corresponding document.

In order to reduce the downtime during migration, the corresponding documents are not read in – instead, the currency field will be filled with the technical currency '2' (indicating two decimals). This ensures a consistent conversion and enables the customer to immediately start working with the data. However, SAP recommends exchanging this provisional solution with the correct document currency. We have provided the report PRC_MIG_POST_PROCESSING, which is to be executed as soon as possible after the down time. Doing so adjusts the currency and the field values where necessary. This works independent of client.

When starting the report, you can ignore the test-mode variable input. Following execution, the report will display the execution log. All logs are also stored in table PRC_MIG_LOG.

Please note that it is always save to rerun the program.

> **Caution**
>
> As mentioned above, the document currency field WAERK in table PRCD_ELEMENTS needs to be filled with the currency of the related document header. It is recommended that you do so in the post-processing step of the migration using report PRC_MIG_POST_PROCESSING. For customers' own document-like objects, a similar conversion report should be developed. See PRC_MIG_POST_PROCESSING for more information about how to adjust all necessary data when exchanging the currency.

### 2.3.1 Further Considerations

In addition to the issues described in the previous chapter, you should also consider the following:

- Check if any of the field extensions of data elements could have an impact on the customer code. Currently, pricing itself does not leverage the field extensions, but this might change in the future. However, SAP does not give any commitment that such an extension will be available in future releases.

   Caution

  Formulas and requirement numbers now have a length of NUMC7. If the formula's include name is to be reconstructed from the number (for example, value formula 999 -> FV64A999), you need to take care to pick up only the last three digits, otherwise the formula call will fail.

- Check if any of your own interfaces, such as BAPI, IDOC, or RFC, are affected by these changes. You should consider introducing compatible structures for these interfaces.

   Note

  Until further notice, pricing will not allow counters (field ZAEHK, data type NUMC3) to exceed a value of 99, to keep the interfaces stable. However, this might change in the future.

   Note

  Interfaces such as BAPIs, IDOCs, RFC enabled function modules used for external communication are kept stable. The fields in the interface keep their former length, prolonged fields are added to the end of the structure (compatible change).

  The following conventions apply for all pricing induced enhancements:

  **Outbound Convention**

  - The long field always contains the correct value.
  - The short field contains the value in case the value fits into the short field.
  - If the value is longer than the short field, the short field stays empty.

  If you are using field content longer than the short field or want to be prepared, you need to adapt your interface call.

  If you do not intend to use longer values (e.g. because you deal with calls to/from classic Suite), the interface call can still use the short field.

  **Inbound Convention**

  - If the long field is filled with valid data by the caller, this value will be taken.
  - If the short field is filled with valid data and the long field is empty, the short field value will be taken.
  - If both are filled but content is not identical, the value for the long fieldname is taken.

  If you are using field content longer than the short field or want to be prepared, you need to adapt your interface call.

  If you do not intend to use longer values (e.g. because you deal with calls to/from classic Suite), the interface call can still use short field.

- Customers need to check if their coding refers to one of the following obsolete fields: KOLNR3, WEGXX, and STUFE. These fields will no longer be stored in Pricing, and you should adjust your coding accordingly.

## 2.4 API Documentation

The new pricing result API is part of package VF_PRC_DB. The package exposes the following relevant parts of the API via the package interface PRC_DB:

### 2.4.1 Factory Class for Accessing Pricing Result Data

The factory class CL_PRC_RESULT_FACTORY is used to retrieve an instance of the actual API.

- Retrieve an instance of this class through the method GET_INSTANCE.
- Use method GET_PRC_RESULT to retrieve the pricing result API interface IF_PRC_RESULT.

### 2.4.2 Interface for Pricing Result Data

The interface for pricing result data (IF_PRC_RESULT) provides API methods to read, write, and delete pricing result data. It is used to decouple the current data persistence from the application code.

> **Note**
>
> Within the SAP Business Suite, the implementation of the interface will execute the current database statements on database table KONV (available from SAP ERP 6.0 EHP 8 onwards).
>
> Within the SAP S/4HANA code line, the interface implementation delegates these operations to the new database table PRCD_ELEMENTS.

The following methods are provided within the API to encapsulate database access:

- GET_PRICE_ELEMENT_DB reads a set of lines from the pricing result by matching a given set of attributes. A catchable exception CX_PRC_RESULT is raised if the request data is invalid.
- GET_PRICE_ELEMENT_DB_BY_KEY reads a specific line from the pricing result by its key. A catchable exception CX_PRC_RESULT is raised if the request data is invalid.
- SAVE_PRICE_ELEMENT_DB writes pricing result data to the database table. As the new database table PRCD_ELEMENTS persists the document currency, it is required to supply the document currency for each pricing document (technical key: KNUMV). A catchable exception CX_PRC_RESULT is raised if the document currency is not provided or if a duplicate key is detected during the insert.

> **Note**
>
> The save method simply tries to insert the provided data into the database table. The method does not provide update-like processing of the provided pricing result set. Hence, for update-like scenarios, you should delete old or outdated data prior to calling the save method. For more information, see the code example section.

> **Note**
>
> The save method does not trigger a database commit because we expect that the pricing result is embedded within a business document, which controls the commit or rollback centrally.

- DELETE_PRICE_ELEMENT_DB_BY_KEY: Deletes a dedicated pricing document or pricing document item.
- DELETE_PRICE_ELEMENT_DB: Supports deletion of multiple pricing documents (list of KNUMVs).
- DELETE_PRICE_ELEMENTS_DB: Delete a list of dedicated pricing result lines.

Caution

Be careful to delete parts of a pricing result, because this might cause inconsistencies. The API does not trigger or enforce any recalculations of pricing documents. It actually really operates on database record level. We provided the methods after reviewing typical access patterns to database table KONV. Nevertheless, if you use the API inappropriately, you may experience side effects.

## 2.5 Code Examples

### 2.5.1 Update on Database Table KONV

The following example illustrates how you can leverage the new API to substitute typical updates on database table KONV. For example, the following code in the include LV45UF0K deletes existing records from database table KONV and inserts new or updated records into the database table.

Syntax

```
FORM KONV_BEARBEITEN.

  CHECK: FKONV_GEAENDERT NE SPACE OR
         VORGANG = CHARH.

  IF VORGANG NE 'H'.
    DELETE FROM KONV WHERE KNUMV = VBAK-KNUMV.
ENHANCEMENT-POINT KONV_BEARBEITEN_10 SPOTS ES_SAPLV45U.
  ENDIF.

  LOOP AT FXKOMV.
    FXKOMV-MANDT = VBAK-MANDT.
    FXKOMV-KNUMV = VBAK-KNUMV.
    MODIFY FXKOMV.
  ENDLOOP.
  INSERT KONV FROM TABLE FXKOMV.
  IF SY-SUBRC NE 0.
    MESSAGE A100 WITH 'KONV' SY-SUBRC.
  ENDIF.

ENHANCEMENT-POINT KONV_BEARBEITEN_11 SPOTS ES_SAPLV45U.
ENDFORM.
```

Using the pricing result AP,I the code is adapted as follows:

Syntax

```
FORM KONV_BEARBEITEN.

  DATA: LT_KONV     TYPE STANDARD TABLE OF KONV,
        LS_DOC_CURR TYPE IF_PRC_RESULT_DATABASE=>TY_DOC_CURRENCY_S,
        LT_DOC_CURR TYPE IF_PRC_RESULT_DATABASE=>TY_DOC_CURRENCY_T.

  CHECK: FKONV_GEAENDERT NE SPACE OR
         VORGANG = CHARH.

  IF VORGANG NE 'H'.
    TRY.
        CL_PRC_RESULT_FACTORY=>GET_INSTANCE( )->GET_PRC_RESULT( )-
>DELETE_PRICE_ELEMENT_DB_BY_KEY(
          EXPORTING
            IV_KNUMV      =     VBAK-KNUMV
        ).
      CATCH CX_PRC_RESULT INTO DATA(LR_EXC).
        MESSAGE X536(VH) WITH LR_EXC->GET_TEXT( ).
```

```
    ENDTRY.
ENHANCEMENT-POINT KONV_BEARBEITEN_10 SPOTS ES_SAPLV45U.
  ENDIF.

  LOOP AT FXKOMV.
    FXKOMV-MANDT = VBAK-MANDT.
    FXKOMV-KNUMV = VBAK-KNUMV.
    MODIFY FXKOMV.
  ENDLOOP.
  MOVE-CORRESPONDING FXKOMV[] TO LT_KONV.

  CLEAR LS_DOC_CURR.
  LS_DOC_CURR-KNUMV = VBAK-KNUMV.
  LS_DOC_CURR-WAERK = VBAK-WAERK.
  INSERT LS_DOC_CURR INTO TABLE LT_DOC_CURR.
  TRY.
      CL_PRC_RESULT_FACTORY=>GET_INSTANCE( )->GET_PRC_RESULT( )-
>SAVE_PRICE_ELEMENT_DB(
        EXPORTING
          IT_PRC_ELEMENT_CLASSIC_FORMAT = LT_KONV
          IT_PRC_ELEMENT_DOC_CURRENCY   = LT_DOC_CURR ).
    CATCH CX_PRC_RESULT INTO DATA(LR_EXC2).
      MESSAGE X535(VH) WITH LR_EXC2->GET_TEXT( ).
  ENDTRY.

ENHANCEMENT-POINT KONV_BEARBEITEN_11 SPOTS ES_SAPLV45U.
ENDFORM.
```

## 2.5.2 Select Data from Database Table KONV

To read pricing result data through the API, the existing code in your application may need to be adjusted according to the following example (include LV61AA11).

Syntax

```
SELECT * INTO TABLE hkomv
         FROM konv
         WHERE knumv = komk-knumv
         ORDER BY PRIMARY KEY.
```

Using the method GET_PRICE_ELEMENT_DB_BY_KEY of interface IF_PRC_RESULT, the select on database table KONV can be replaced as follows:

Syntax

```
TRY.
    cl_prc_result_factory=>get_instance( )->get_prc_result( )-
      >get_price_element_db_by_key(
        EXPORTING
          iv_knumv                      = komk-knumv
        IMPORTING
          et_prc_element_classic_format = hkomv ).
  CATCH cx_prc_result ##NO_HANDLER. "implement suitable error handling
ENDTRY.
```

Here is an alternative solution to read the same set of data.

Syntax

```
TRY.
    cl_prc_result_factory=>get_instance( )->get_prc_result( )-
      >get_price_element_db(
        EXPORTING
          it_selection_attribute        = VALUE #( ( fieldname = 'KNUMV'
                                                     value     = komk-knumv ) )
        IMPORTING
          et_prc_element_classic_format = hkomv ).
  CATCH cx_prc_result ##NO_HANDLER. "implement suitable error handling
ENDTRY.
```

## 2.5.3 Deleting Data from Database Table KONV

Apart from updates on the KONV table (in which case you first need to delete records before you insert them) there are use cases where you simply need to delete data from database table KONV. To prepare the switch to the new database table PRCD_ELEMENTS you can trigger these deletions using an API method. The following snippet from report SDVFKKDL (deletion report for archiving) illustrates how the adjusted code could look. Prior to the adjustment, data records where deleted from KONV as depicted.

Syntax

```
*---------------------------------------------------------------------*
*  FORM DELETE_KONV
*---------------------------------------------------------------------*
FORM DELETE_KONV.
  DESCRIBE TABLE XKONV LINES LINES.
  CHECK LINES > 0.

  DELETE KONV FROM TABLE XKONV.
*****************************************
* Anfang Änderung Statistikbaustein 03.01.02
*****************************************
  ARCH_STAT-TABNAME = 'KONV'.
  ARCH_STAT-COUNT = sy-dbcnt.
  APPEND ARCH_STAT.
*****************************************
* Ende Änderung Statistikbaustein 03.01.02
*****************************************
  IF SY-SUBRC NE 0.
    TXT1 = 'KONV'.
*   "deletion of table &2 was not or only partially successfull."
    PERFORM PR_FL USING XVFKK-FKNUM SPACE '446' TXT1 SPACE.
  ENDIF.
ENDFORM.
```

And here is the version that leverages the new API:

Syntax

```
FORM DELETE_KONV.
  DESCRIBE TABLE XKONV LINES LINES.
  CHECK LINES > 0.

  TRY.
      cl_prc_result_factory=>get_instance( )->get_prc_result( )-
>delete_price_elements_db(
        EXPORTING
          it_prc_element_classic_format = xkonv[] ).
*****************************************
* Anfang Änderung Statistikbaustein 03.01.02
*****************************************
      ARCH_STAT-TABNAME = 'PRCD_ELEMENTS'.
      ARCH_STAT-COUNT = LINES.
      APPEND ARCH_STAT.
*****************************************
* Ende Änderung Statistikbaustein 03.01.02
*****************************************
    CATCH cx_prc_result.
      TXT1 = 'PRCD_ELEMENTS'.
*   "deletion of table &2 was not or only partially successfull."
      PERFORM PR_FL USING XVFKK-FKNUM SPACE '446' TXT1 SPACE.
  ENDTRY.
ENDFORM.
```

# 3 Changes in Condition Technique

## 3.1 Changes in the Data Model

The concatenated variable key field VAKEY of a condition table has been removed from all condition header tables including KONH (Pricing), NACH (Output determination), KOND3 (Campaign determination), KONDN (Free goods determination), KONHM (Portfolio determination), J_3GPRLHD (CEM price list determination) and WIND (Document index). The concatenated variable data field VADAT has also been removed.. This was done to avoid data migration of these tables as a result of field length extension as, for example, the material number field length extension from CHAR18 to CHAR40.

For internal processing, long data elements VAKEY_LONG and VADAT_KO_LONG with length CHAR255 have been introduced.

The content of the new long VAKEY and VADAT can be determined at runtime with the help of the methods of service class CL_COND_VAKEY_SRV.

For compatibility reasons, the field VAKEY_LONG (CHAR255) was added to the batch input structures (program RV14BTCI). If this field is filled or used by an external program, the content of this field is used instead of the content of the still existing field VAKEY.

For compatibility reasons, the fields MATNR_LONG (CHAR40), UPMAT_LONG (CHAR40), BOMAT_LONG (CHAR40) and VAKEY_255, as well as VADAT_255, have been added to the COND_A IDOC segments. If these fields for the long material number or long VAKEY/VADAT are filled in addition to the still existing short fields, the content of the long fields has priority.

The maximum number of possible accesses in an access sequence (DTEL KOLNR) has been enhanced from 99 to 999. Therefore, the solution described in Pilot SAP Note 1812828 (for customers for which this note was released) is no longer necessary, nor valid. Any content from this solution will be automatically transferred to the standard tables during the migration.

Additional changes: data element QUDIW (special value source in access sequence) has been extended from CHAR20 to CHAR50. Data element SKONDTAB (CHAR128, for archiving SD conditions) has been replaced by SDKONTAB_LONG (CHAR512). Data element KOND_DATA (used for general condition determination) has been extended from CHAR255 to CHAR512.

## 3.2 Code Examples

The fields VAKEY and VADAT, for example those of table KONH, can no longer be accessed directly. Instead, they can be reconstructed from the condition record number and the assigned A-Table content. If you need to access these fields in your coding, you can use methods of the service class CL_COND_VAKEY_SRV. See, for example, the function module RV_CONDITION_RECORD:

Syntax

```
  ELSE.
    SELECT SINGLE * FROM KONH WHERE KNUMH = CONDITION_NUMBER.
  ENDIF.
* record exists
  IF SY-SUBRC <> 0.
```

```
        MESSAGE E058 WITH CONDITION_NUMBER RAISING NO_EXISTING_RECORD.
      ENDIF.
    * Determine access program
  CALL FUNCTION 'RV_CONDITION_MAINTENANCE'
       EXPORTING
            APPLICATION       = KONH-KAPPL
            CONDITION_TABLE   = KONH-KOTABNR
            CONDITION_TYPE    = KONH-KSCHL
            KEY_FIELDS        = KOMG
            VARIABLE_KEY      = KONH-VAKEY
            MAINTAIN_MODE     = MAINTAIN_MODE
            FIRST_SCREEN      = FIRST_SCREEN
            RECORD_NUMBER     = KONH-KNUMH
```

And here is the version that leverages the new API:

Syntax

```
 data: lv_vakey type vakey_long,
        lo_cond_vakey_srv type ref to cl_cond_vakey_srv,
        o_cx_ref type ref to cx_cond_vakey.
……
 ELSE.
   SELECT SINGLE * FROM KONH WHERE KNUMH = CONDITION_NUMBER.
   if sy-subrc = 0.
     try.
       lv_vakey = lo_cond_vakey_srv->determine_vakey_from_db( iv_usage   = konh-kvewe
                                                              iv_knumh   = konh-knumh
                                                             kotabnr = konh-kotabnr ).
      catch cx_cond_vakey into o_cx_ref.
* some error handling if needed
         clear lv_vakey.
*        sy-subrc = 4.
      endtry.
    endif.
  ENDIF.
* record exists
  IF SY-SUBRC <> 0.
    MESSAGE E058 WITH CONDITION_NUMBER RAISING NO_EXISTING_RECORD.
  ENDIF.
* Determine access program
  CALL FUNCTION 'RV_CONDITION_MAINTENANCE'
       EXPORTING
            APPLICATION       = KONH-KAPPL
            CONDITION_TABLE   = KONH-KOTABNR
            CONDITION_TYPE    = KONH-KSCHL
            KEY_FIELDS        = KOMG
            VARIABLE_KEY      = LV_VAKEY
```

# 4 Industry Specifications

## 4.1 IS-OIL Specific Actions:

- Several data model changes in the Pricing and Condition Technique area are provided with SAP S/4HANA Oil and Gas 1610. These are:
- Replacement of table KONV with the new table PRCD_ELEMENTS (Usage of API methods and CDS Views) as data persistency for pricing results. KONV is and can still be used for data declaration purposes. It still defines the structure of the pricing result within the application code.
- The concatenated variable key field VAKEY of a condition table has been removed from all condition header tables, including KONH (pricing), NACH (output determination), KOND3 (campaign determination), KONDN (free goods determination), KONHM (portfolio determination), the field VADAT has also been removed.
- Several field length extensions like ZAEHK – Crated new field ZAEHK_LONG in structures which are used by RFCs (assuming user might use CHAR3 or CHAR2 according to their usage) :
    - OICQ7_KEY
    - OICQ8_KEY
    - OICQ9_KEY

SAP also made changes to the RFCs which are using the above structures.

OICQ7_READ_OB
OICQ8_PUT_DB
OICQ8_PUT_OB
OIC_CALL_MAINTAIN_FORMULA_UI
OIC_DOCFORMULA_ALL_SAVE_OB
OIC_SET_TRAFFIC_LIGHT
OICQ8_READ_OB
OICQ8_SAVE_OB
OICQ9_PUT_DB
OICQ9_SAVE_OB
OICQ7_PUT_DB
OICQ7_PUT_OB
OICQ7_READ_DB
OICQ7_READ_OB
OICQ7_SAVE_OB
OICQ8_READ_OB
OICQ8_SAVE_OB
OICQ9_READ_OB
OICQ9_SAVE_OB
OIC_CALL_MAINTAIN_FORMULA_UI
OIC_DOCFORMULA_ALL_SAVE_OB

Since the changes are already made, no action is expected from IS-OIL customer

Oil appends (OI0_PRCD_APP) were attached to the structure PRCS_ELEMENTS_DATA which is available in the tables mentioned below.

List of table:

PRCD_ELEMENTS

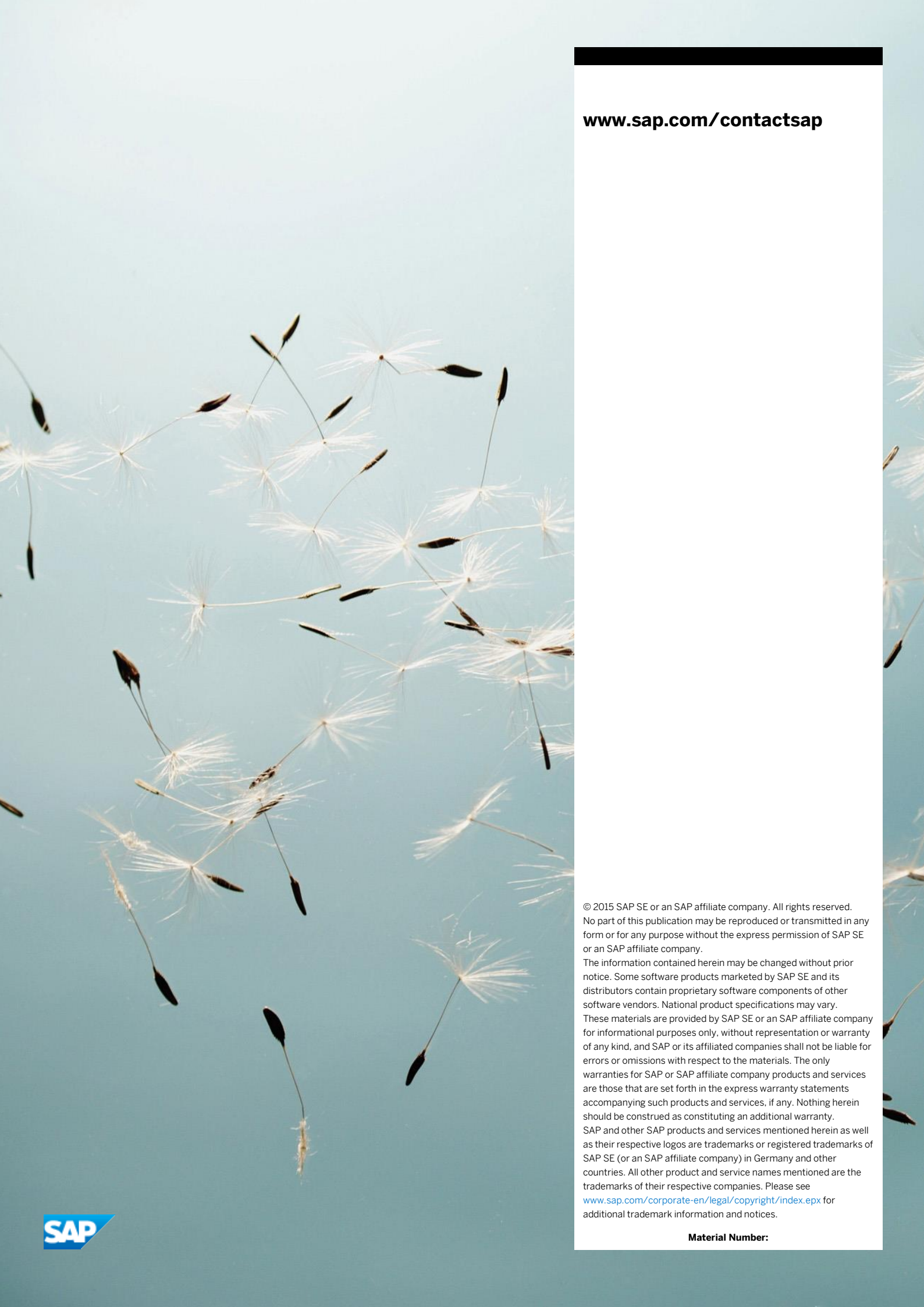
PRCD_ELEM_DRAFT

List of fields in oil append OI0_PRCD_APP:

| Name of the Field | Type of the Field | Length | Description |
|---|---|---|---|
| OIKOSTL | CHAR | 10 | Cost center for fee expense accounting |
| OIACPOL | CHAR | 1 | Fee accounting policy indicator |
| OIFEEMAN | CHAR | 1 | Fee rate indicator |
| OIKWERT | CURR | 13 | Fee detail clearing amount |
| OIFEEPRT | CHAR | 1 | Condition is printed at item level |
| OIGRNET | CHAR | 1 | Gross/net pricing indicator |
| OIGNSTRA | CHAR | 1 | Gross/net pricing strategy used |
| OIGRCHK | CHAR | 1 | Carry  out gross/net unit check |
| OIINVCYC | NUMC | 1 | Invoice cycle |
| OIINVCYACT | CHAR | 1 | Invoice cycle included indicator active (X/ ) |
| OIINCYST | CHAR | 1 | Cond. statistical due to invoicing cycle |
| OIA_SPLTIV | CHAR | 1 | Indicator for split invoice verification |
| OIA_ZTERM | CHAR | 4 | Terms of payment key |
| OILICIN | CHAR | 10 | Excise tax internal license number |
| OILICTP | CHAR | 4 | License type |
| OIREPORT | CHAR | 8 | Report number |
| OIDATA | NUMC | 3 | Routine number for data capture |
| OIHEAD | NUMC | 3 | Header format routine for second level analysis report |
| OIITEM | NUMC | 3 | Item routine for second level analysis report |
| OIERRHAN | NUMC | 3 | Error handling routine for second level analysis |
| OIU_EP | CHAR | 1 | Exploration and Production condition (PRA) |
| OIRCONDCL | CHAR | 1 | SSR PC: Condition classification |

.

The CDS views V_KONV and V_KONV_DRAFT are extended with the help of report ROIL_CDS_EXTENSION.



**www.sap.com/contactsap**



**Material Number:**