SAP®

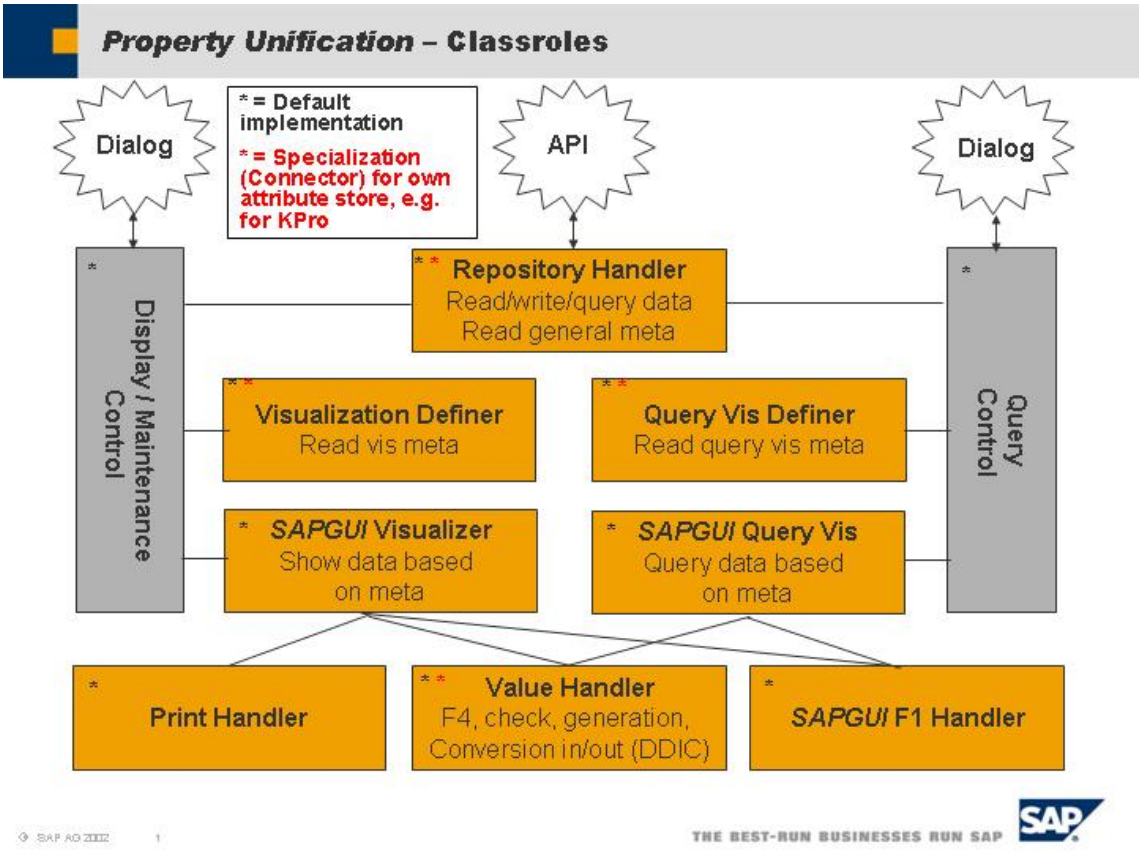
Cross SP search based on Property Unification

Contents

Introduction 1
Covered functionality 2
Customizing capabilities 4
Introducing new aliases 6
Introducing new SPs and new elementtypes 8
Datatype conversions 8
Technical solution 9

Introduction

The Property Unification (short PU) defines a set of generic (SP independent) classroles and interfaces for attribute data and metadata access. PU forces a clean separation between backend, frontend and controller logic and PU forces a clean separation between data and metadata handling. Each SP is able to connect to PU by implementing mandatory PU classroles (interfaces) or by re-using the default implementations.



The cross SP search solution (short CRQY) benefits from this PU abstraction concept. CRQY utilizes existing PU classroles to read SP specific attribute metadata and to start SP specific queries.

The main idea behind CRQY solution is aliazing of attribute IDs: each attribute (attribute description object, served by RCM framework) has an unique ID and an optional ALIAS_ID. This solves the mapping problem between CRQY and SP metadata.

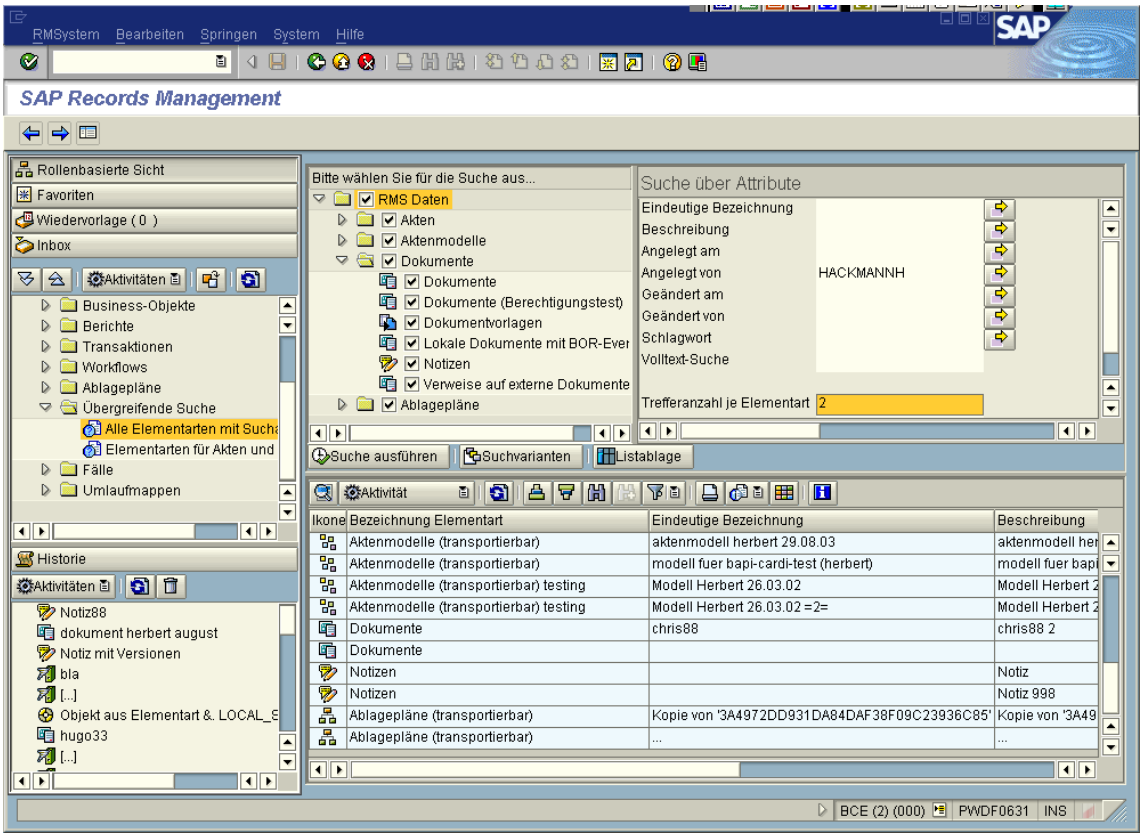
I introduced new classroles for CRQY itself, since i learned from history, that stakeholders/partners/customers want to specialize anything in frontend, backend and controller logic ;-). With the finegranular CRQY classroles they able to do so.

Covered functionality

The CRQY solution offers a SAPGUI control. This control can be called inplace and outplace. The report SRM_PROP_CROSS_QUERY_HOWTO shows an example for calling CRQY control outplace.

The CRQY solution also offers a “dark” cross query API, so that API based RCM solutions (e.g. CRM) could use CRQY too. The report SRM_PROP_CROSS_QUERY_HOWTO shows an calling example for a “dark” search on all CRQY searchable elementtypes.

The CRQY control is called inplace by the new service provider SRM_SP_QUERY (SCMG_SP_QUERY), the corresponding new elementtype SRM_SPS_QUERY (SCMG_SPS_QUERY) is shown in organizer (e.g. in role based tab). Doubleclick on elementtype SRM_SPS_QUERY and start of selection leads to the following search screen…



As known from single SP search - the selection screen (including checked elementtypes) and the hitlist can be saved and reopenend.

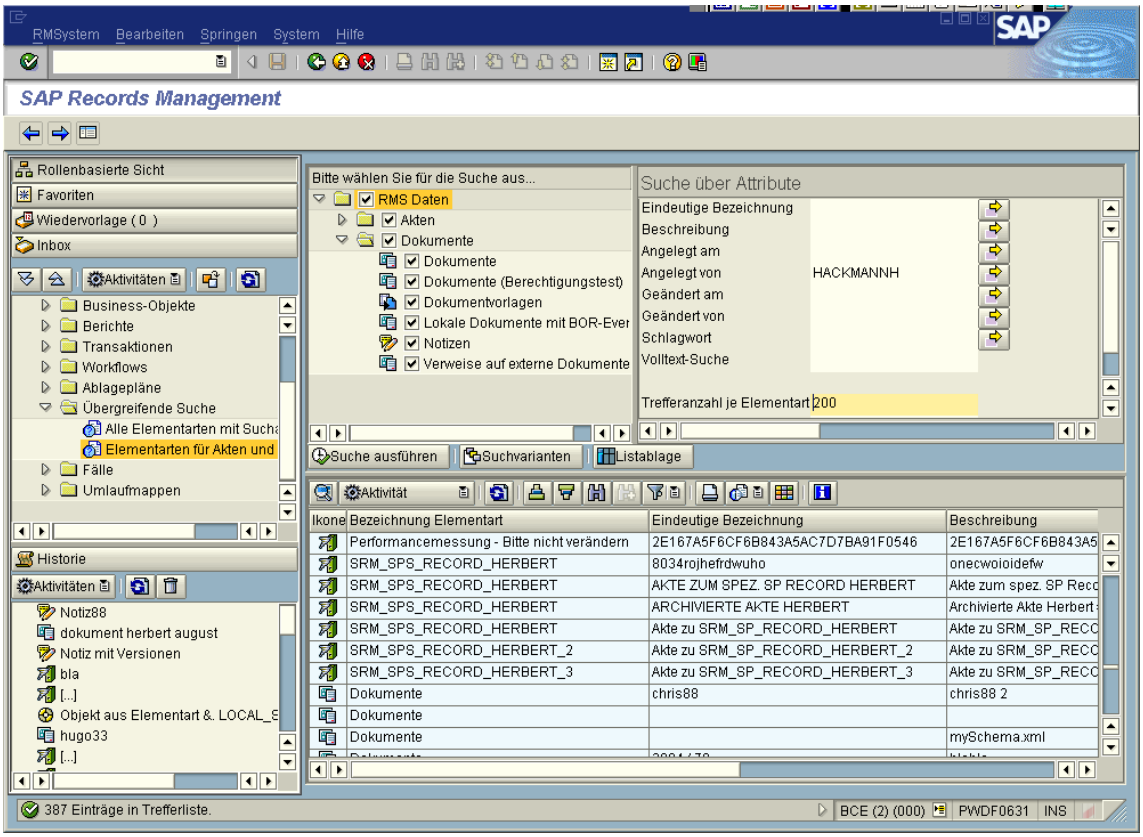
From customizing point of view, it's possible to setup/change the list of searchable SP types and elementtypes, the list of attributes in selection screen and the list of attributes in hitlist. It's also possible to define in customizing all metadata (including visualization details like row/column number for selection screen and column number for hitlist) for each CRQY attribute. As known from PU - special (non DDIC) value help/check can be connected to each CRQY attribute in selection screen. Please have a look at section Customizing capabilities for details.

There's also a dummy attribute defined/delivered which realizes a fulltext search field, so that a easy fulltext search functionality is covered too (analogous to single SP search).

The checkboxes at elementtype selection tree are somehow "intelligent": Check of a box checks all tree children too, uncheck of a box unchecks all tree children too. Moreover when a tree node is checked, it's checked, if nodes in the parent chain can be checked too (when all children are checked). If you don't understand this explanation, make a try ;-).

The list of searchable elementtypes is determined at runtime via a PU interface (SP implements participation method). This dynamic list can (but doesn't have to) be filtered by SP types and/or elementtypes in customizing: The SP SRM_SP_QUERY (SCMG_SP_QUERY) is delivered with two elementtypes: the first one SRM_SPS_QUERY (SCMG_SPS_QUERY) filters nothing and covers all searchable elementtypes and the second one SRM_SPS_QUERY_REC_DOC

(SCMG_SPS_QUERY_REC_DOC) covers all searchable elementtypes with SP type SRM_RECORD and SRM_DOCUMENT. Doubleclick on elementtype SRM_SPS_QUERY_REC_DOC and start of selection leads to the following search screen…

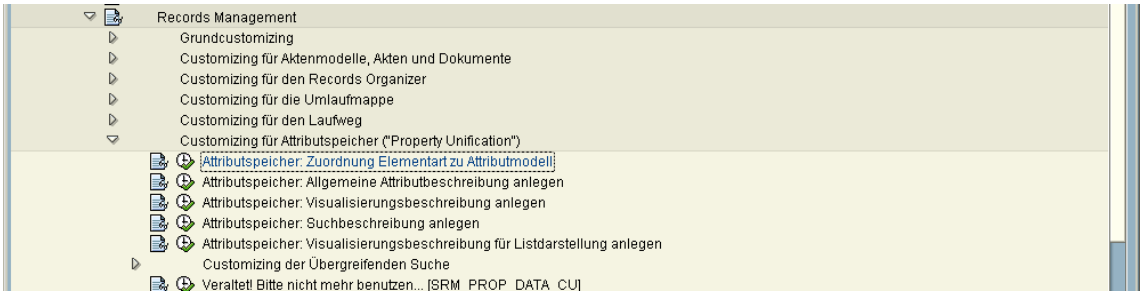


Customizing capabilities

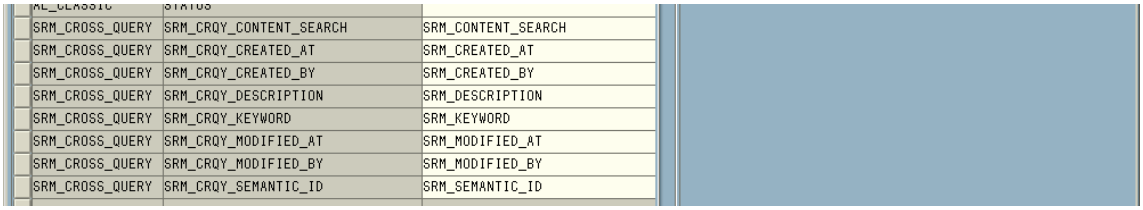
As already mentioned, there're several customizing capabilities: it's possible to setup/change the list of searchable SP types and elementtypes, the list of attributes in selection screen and the list of attributes in hitlist. It's also possible to define in customizing all metadata for each CRQY attribute.

Customizing of attributes in selection screen and hitlist

There're some new PU customizing views in transaction SPRO. These views are used for PU default attribute repository customizing and for CRQY attribute customizing…

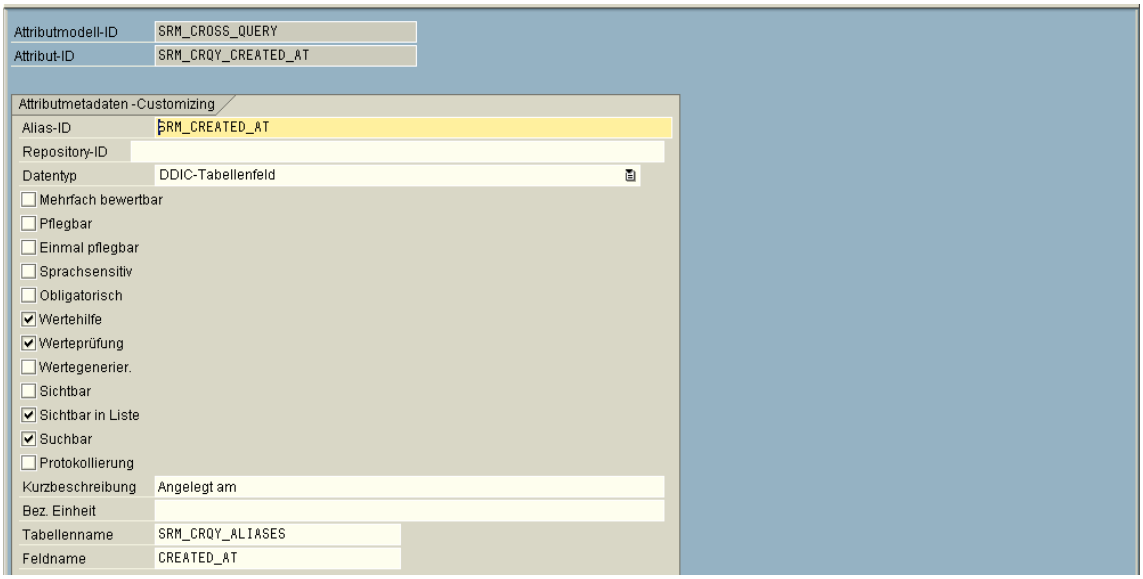


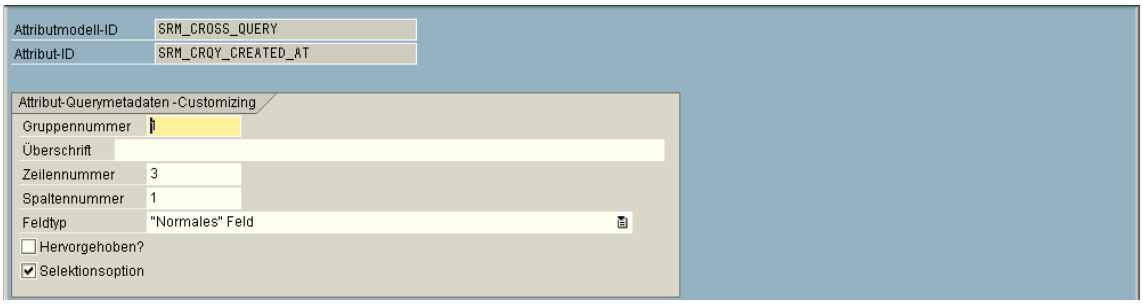
We deliver a standard attributmodel SRM_CROSS_QUERY (grouping of attributes) and a standard set of attributes SRM_CRQY_* (with aliases SRM_*). This attributmodel is assigned to our delivered elementtypes SRM_SPS_QUERY* (SCMG_SPS_QUERY*).



SRM_CROSS_QUERY	SRM_CRQY_CONTENT_SEARCH	SRM_CONTENT_SEARCH
SRM_CROSS_QUERY	SRM_CRQY_CREATED_AT	SRM_CREATED_AT
SRM_CROSS_QUERY	SRM_CRQY_CREATED_BY	SRM_CREATED_BY
SRM_CROSS_QUERY	SRM_CRQY_DESCRIPTION	SRM_DESCRIPTION
SRM_CROSS_QUERY	SRM_CRQY_KEYWORD	SRM_KEYWORD
SRM_CROSS_QUERY	SRM_CRQY_MODIFIED_AT	SRM_MODIFIED_AT
SRM_CROSS_QUERY	SRM_CRQY_MODIFIED_BY	SRM_MODIFIED_BY
SRM_CROSS_QUERY	SRM_CRQY_SEMANTIC_ID	SRM_SEMANTIC_ID

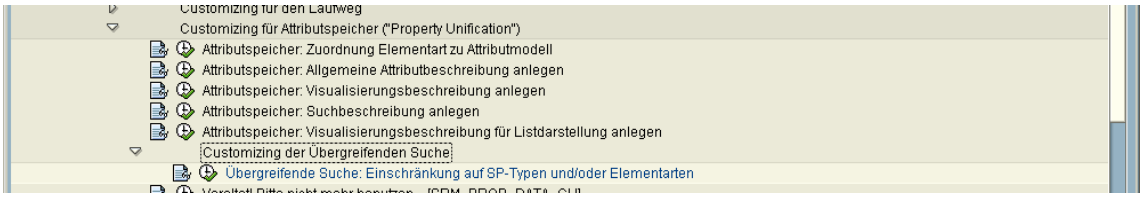
For instance the attribute metadata customizing (general, query visualization and hitlist visualization) for attribute SRM_CRQY_CREATED_AT looks like following…



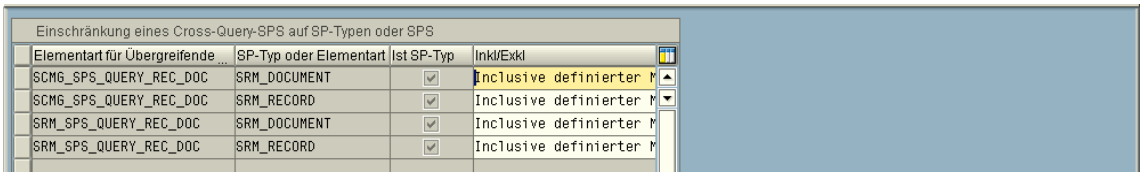


<u>Customizing of SP types and elementtypes</u>

The list of searchable elementtypes is determined at runtime via a PU interface (SP implements participation method). This dynamic list can (but doesn't have to) be filtered by SP types and/or elementtypes. There's a new CRQY customizing view in transaction SPRO…



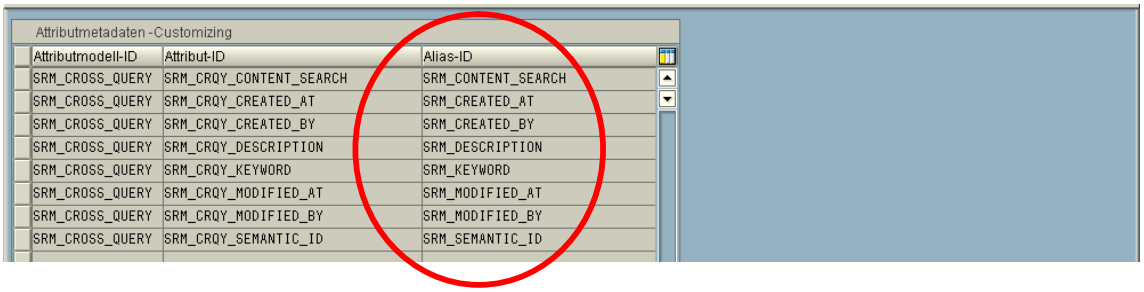
For instance you'll find an restriction of elementtype SRM_SPS_QUERY_REC_DOC (SCMG_SPS_QUERY_REC_DOC) to SP types SRM_DOCUMENT and SRM_RECORD…



There're some limitations to customizing combinations (mixing up of including/excluding across SP type sets and elementtyp sets), you'll find them in the documentation, which is attached to the customizing node in transaction SPRO.

Introducing new aliases

Maybe stakeholders/partners/customers are not fine with the delivered list of CRQY aliases SRM_CRQY_*…



Therefore it's possible - as already mentioned in section Customizing capabilities - to introduce new CRQY attributes and/or aliases. But the interesting point here is, how to introduce these new aliases to the SP specific metadata, so that the mapping between CRQY and SP attribute IDs can be done.

There're (in general) 2 ways to introduce new aliases to SP specific metadata…

1. Attach aliases to attribute IDs in SP specific customizing
2. Attach aliases to attribute IDs in implementation of PU repository classrole

The first way is – of course – the nicer way, but for some SP this might not be possible. The second way is for any SP possible.

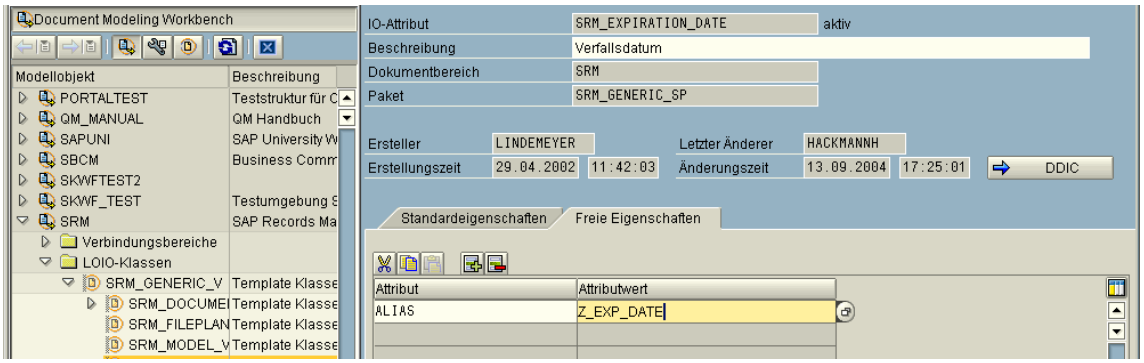
SP using the PU default attribute repository (e.g. SP AL in 7.0)

There're both ways possible, since the customizing views for the definition of the attribute metadata allow the specification of alias IDs.

By the way: the customizing views for the PU default attribute repository are the same than for CRQY attribute definition, please have a look at section Customizing capabilities.

KPro based SPs (SP Record, SP Document etc.)

For SP, which store there're attributes in KPro, there're both ways possible. It's possible to set a value for special free attribute ALIAS directly in transaction DMWB and this value is recognized in KPro implementation of PU repository classrole…



It's also possible to specialize directly KPro implementation of PU repository classrole. Therefore I would recommend to specialize method CL_SRM_GSP_PROP_REPOSITORY-> GSP_ALIAS_GET…

```
method GSP_ALIAS_GET.
  data myNameValuePair type line of srmgspropt.

  case id.
    when if_srm_document=>prop_document_id.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_SEMANTIC_ID.
    when if_srm_document=>prop_description.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_DESCRIPTION.
    when if_srm_document=>prop_created_at.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_CREATED_AT.
    when if_srm_document=>prop_created_by.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_CREATED_BY.
    when if_srm_document=>prop_content_changed_at.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_MODIFIED_AT.
    when if_srm_document=>prop_content_changed_by.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_MODIFIED_BY.
    when if_srm_document=>prop_keyword.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_KEYWORD.
    when IF_SRM_SRM_PROP_QUERY_DEFINE=>PROP_ID_CONTENT_SEARCH.
*     support standard aliases...
      alias = IF_SRM_SRM_PROP_CROSS_QUERY=>ALIAS_CONTENT_SEARCH.
    when others.
*     support additional custom aliases...
```

```
      if property_type_wa is initial.
        alias = ''.
      else.
        read table property_type_wa-additional_props with key name = 'ALIAS'
          into myNameValuePair.
        if sy-subrc = 0.
          alias = myNameValuePair-value.
        else.
          alias = ''.
        endif.
      endif.
  endcase.
endmethod.
```

Introducing new SPs and new elementtypes

The precondition for the connection to CRQY is the connection to PU: GSP based SPs are automatically connected and there's also the possibility to reuse the PU default repository implementation.

Additionally the classrole IS_SP_PROP_CROSS_QUERY has to be implemented and registered in SRMREGEDIT. This is easy, since there's only to implement method PARTICIPATE by returning IF_SRM=>TRUE.

New elementtypes are automatically recognized when the corresponding SP is recognized.

Datatype conversions

Sometimes there's a need for conversion of datatypes. E.g. there's actually a mismatch regarding attributes SRM_CRQY_CREATED_AT/SRM_CRQY_MODIFIED_AT between CRQY and SP metadata definition. In CRQY the attributes are declared as type DATE and in SP RECORD/DOCUMENT as type TIMESTAMP.

The SP specific PU implementation has to deal with this and has to make the necessary conversions! The SP can determine via context, if query is started in a CRQY scenario.

E.g. this is done for GSP PU implementation in method CL_SRM_GSP_PROP_REPOSITORY ->IF_SRM_SP_PROP_REPOSITORY~QUERY …

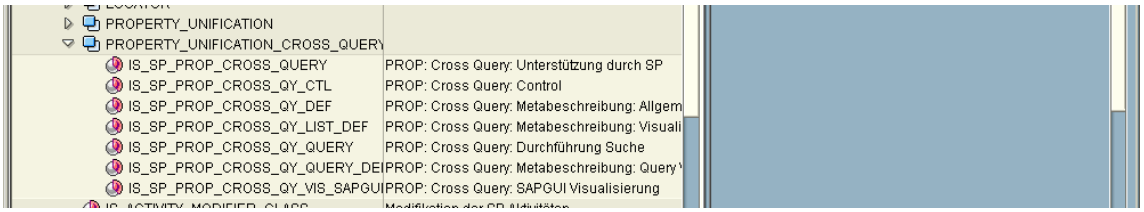
```
*========== cross query: conversion from DATS to TIMESTAMP needed ==========*
      if query_context-CALLED_BY_CROSS_QUERY = if_srm=>true.
        data lt_query_wa type line of sdm_query_desc_tab.
        loop at lt_query_tab into lt_query_wa.
          if lt_query_wa-property_id = 'CREATED_AT'
            or lt_query_wa-property_id = 'LAST_CHANGED_AT'.
            me->DATE_2_TIMESTAMP( changing query_wa = lt_query_wa ).
            modify lt_query_tab from lt_query_wa transporting operator value
high_value.
          endif.
        endloop.
      endif.
*========== cross query: conversion from DATS to TIMESTAMP needed ==========*
```

Technical solution

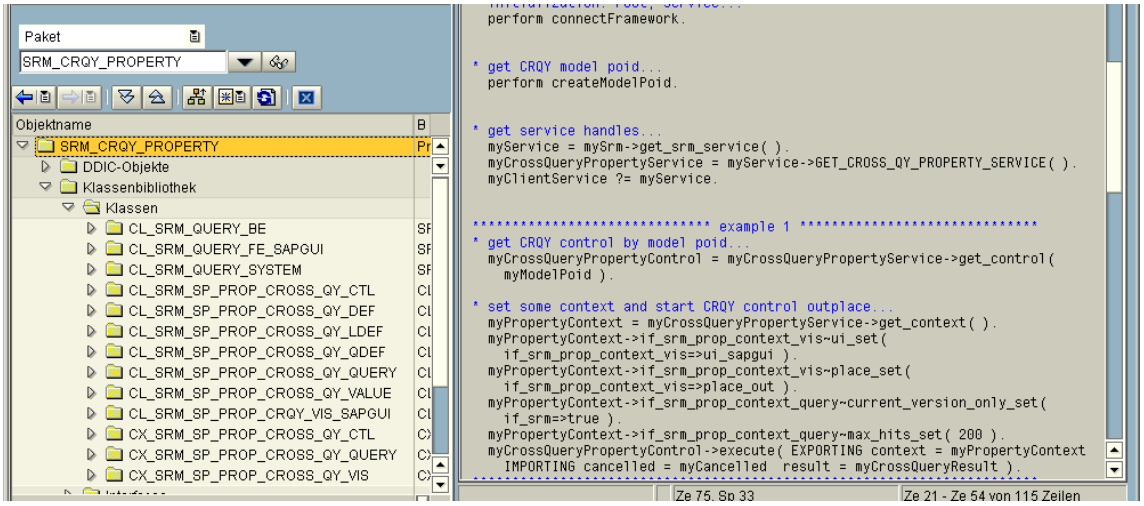
CRQY utilizes existing PU classroles to read SP specific attribute metadata and to start SP specific queries.

The main idea behind CRQY solution is aliazing of attribute IDs: each attribute (attribute description object, served by RCM framework) has an unique ID and an optional ALIAS_ID. This solves the mapping problem between CRQY and SP metadata.

CRQY solution consists of the implementation and co-work of 7 new classroles…



In package SRM_CRQY_PROPERTY you'll find an implementation for each classrole except for IS_SP_PROP_CROSS_QUERY, since this classrole (participation classrole) has to be implemented by the SPs, which want to take part at CRQY search.



I don't want to describe the classroles, interfaces and classes of package SRM_CRQY_PROPERTY in detail. This should be clear by looking at the entities itself (tricky coding parts are introduced by comments!).

With the finegranular CRQY classroles stakeholders/partners/customers are able to specialize nearly anything in backend, frontend and controller logic.