SAP DEVELOPER NETWORK

SDN Community Contribution

(This is not an official SAP document.)

KM Web Services

Applies To:

This article applies to Knowledge Management in the next SAP NetWeaver release.

Summary

The article gives a sneak preview of the new Web services for Knowledge Management & Collaboration (KMC), which are part of the next SAP NetWeaver release.

By: Vedran Lerenc

Company: SAP AG

Date: 18.11.2005

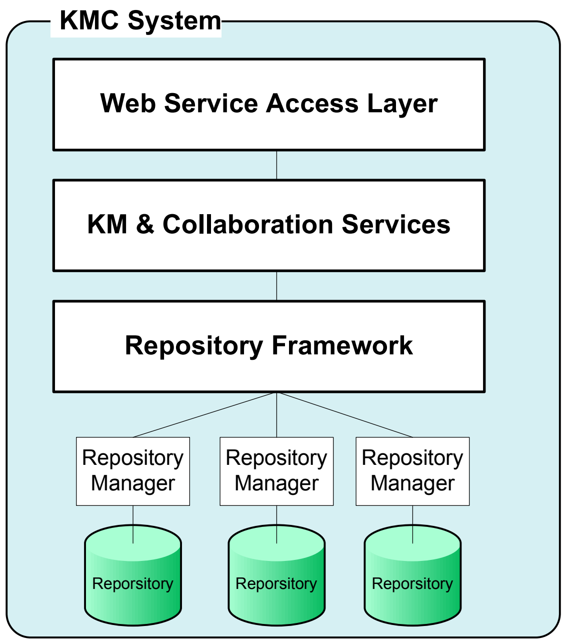
Table of Contents

Introduction

With Knowledge Management (KM) in the next SAP NetWeaver release, we provide Web services on top of KM, which allow you to use Knowledge Management not only on the Java platform, but on any Web service-enabled platform in a remote scenario based on SOAP. The first version of the new Web services includes the functions of the repository framework (core KM function) and access to the KM index management service (KM search function). More services will follow in the future – Web services for Collaboration Rooms are already planned.

Overview

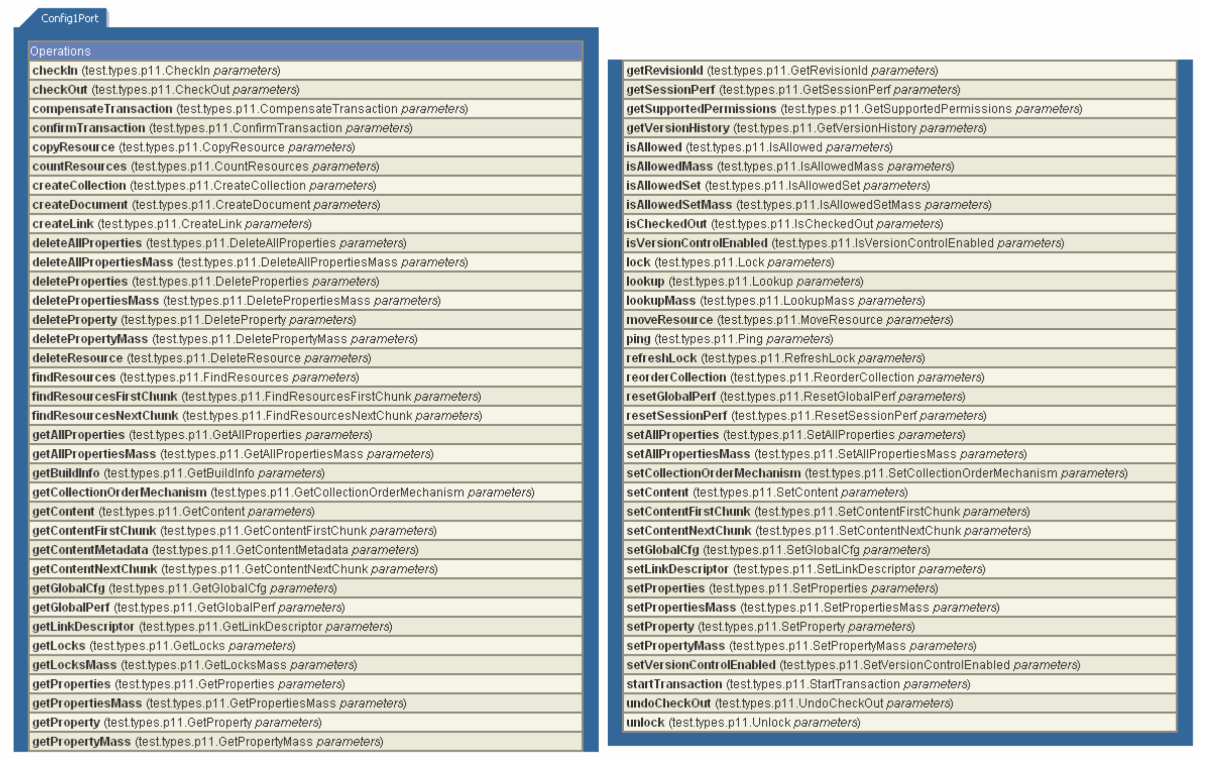
With its repository framework and KMC services, KMC runs on a J2EE Engine. On top of this, KMC provides a Web service access layer that allows access to the repository framework and KMC services from any platform enabled for SOAP. You could even use KM Web services from a .NET platform!



The Web services have been built using SAP’s Enterprise Service Infrastructure (ESI), the core of SAP’s Enterprise Service Architecture (ESA). They were modeled and then generated. This procedure spared us a lot of tedious and error-prone work and allowed us to concentrate on the real implementation, bridging from the Web services to KM and back. The heart of the implementation is a Java Bean, which is called by ESI at runtime.

API

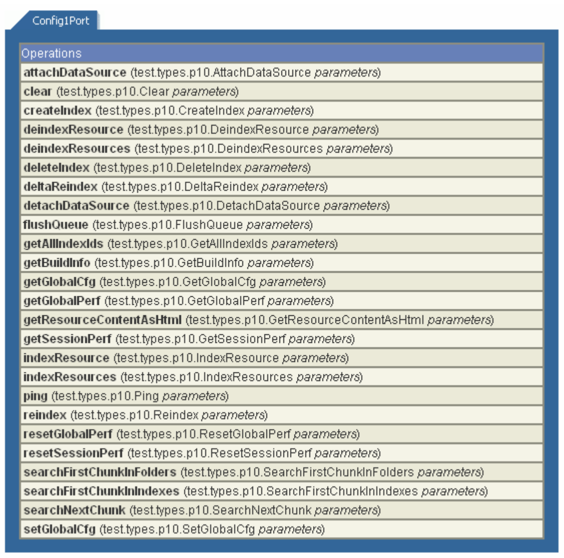
You can find below an overview of the Web service methods currently supported for the repository framework:



Operations
checkIn (test.types.p11.CheckIn *parameters*)
checkOut (test.types.p11.CheckOut *parameters*)
compensateTransaction (test.types.p11.CompensateTransaction *parameters*)
confirmTransaction (test.types.p11.ConfirmTransaction *parameters*)
copyResource (test.types.p11.CopyResource *parameters*)
countResources (test.types.p11.CountResources *parameters*)
createCollection (test.types.p11.CreateCollection *parameters*)
createDocument (test.types.p11.CreateDocument *parameters*)
createLink (test.types.p11.CreateLink *parameters*)
deleteAllProperties (test.types.p11.DeleteAllProperties *parameters*)
deleteAllPropertiesMass (test.types.p11.DeleteAllPropertiesMass *parameters*)
deleteProperties (test.types.p11.DeleteProperties *parameters*)
deletePropertiesMass (test.types.p11.DeletePropertiesMass *parameters*)
deleteProperty (test.types.p11.DeleteProperty *parameters*)
deletePropertyMass (test.types.p11.DeletePropertyMass *parameters*)
deleteResource (test.types.p11.DeleteResource *parameters*)
findResources (test.types.p11.FindResources *parameters*)
findResourcesFirstChunk (test.types.p11.FindResourcesFirstChunk *parameters*)
findResourcesNextChunk (test.types.p11.FindResourcesNextChunk *parameters*)
getAllProperties (test.types.p11.GetAllProperties *parameters*)
getAllPropertiesMass (test.types.p11.GetAllPropertiesMass *parameters*)
getBuildInfo (test.types.p11.GetBuildInfo *parameters*)
getCollectionOrderMechanism (test.types.p11.GetCollectionOrderMechanism *parameters*)
getContent (test.types.p11.GetContent *parameters*)
getContentFirstChunk (test.types.p11.GetContentFirstChunk *parameters*)
getContentMetadata (test.types.p11.GetContentMetadata *parameters*)
getContentNextChunk (test.types.p11.GetContentNextChunk *parameters*)
getGlobalCfg (test.types.p11.GetGlobalCfg *parameters*)
getGlobalPerf (test.types.p11.GetGlobalPerf *parameters*)
getLinkDescriptor (test.types.p11.GetLinkDescriptor *parameters*)
getLocks (test.types.p11.GetLocks *parameters*)
getLocksMass (test.types.p11.GetLocksMass *parameters*)
getProperties (test.types.p11.GetProperties *parameters*)
getPropertiesMass (test.types.p11.GetPropertiesMass *parameters*)
getProperty (test.types.p11.GetProperty *parameters*)
getPropertyMass (test.types.p11.GetPropertyMass *parameters*)
getRevisionId (test.types.p11.GetRevisionId *parameters*)
getSessionPerf (test.types.p11.GetSessionPerf *parameters*)
getSupportedPermissions (test.types.p11.GetSupportedPermissions *parameters*)
getVersionHistory (test.types.p11.GetVersionHistory *parameters*)
isAllowed (test.types.p11.IsAllowed *parameters*)
isAllowedMass (test.types.p11.IsAllowedMass *parameters*)
isAllowedSet (test.types.p11.IsAllowedSet *parameters*)
isAllowedSetMass (test.types.p11.IsAllowedSetMass *parameters*)
isCheckedOut (test.types.p11.IsCheckedOut *parameters*)
isVersionControlEnabled (test.types.p11.IsVersionControlEnabled *parameters*)
lock (test.types.p11.Lock *parameters*)
lookup (test.types.p11.Lookup *parameters*)
lookupMass (test.types.p11.LookupMass *parameters*)
moveResource (test.types.p11.MoveResource *parameters*)
ping (test.types.p11.Ping *parameters*)
refreshLock (test.types.p11.RefreshLock *parameters*)
reorderCollection (test.types.p11.ReorderCollection *parameters*)
resetGlobalPerf (test.types.p11.ResetGlobalPerf *parameters*)
resetSessionPerf (test.types.p11.ResetSessionPerf *parameters*)
setAllProperties (test.types.p11.SetAllProperties *parameters*)
setAllPropertiesMass (test.types.p11.SetAllPropertiesMass *parameters*)
setCollectionOrderMechanism (test.types.p11.SetCollectionOrderMechanism *parameters*)
setContent (test.types.p11.SetContent *parameters*)
setContentFirstChunk (test.types.p11.SetContentFirstChunk *parameters*)
setContentNextChunk (test.types.p11.SetContentNextChunk *parameters*)
setGlobalCfg (test.types.p11.SetGlobalCfg *parameters*)
setLinkDescriptor (test.types.p11.SetLinkDescriptor *parameters*)
setProperties (test.types.p11.SetProperties *parameters*)
setPropertiesMass (test.types.p11.SetPropertiesMass *parameters*)
setProperty (test.types.p11.SetProperty *parameters*)
setPropertyMass (test.types.p11.SetPropertyMass *parameters*)
setVersionControlEnabled (test.types.p11.SetVersionControlEnabled *parameters*)
startTransaction (test.types.p11.StartTransaction *parameters*)
undoCheckOut (test.types.p11.UndoCheckOut *parameters*)
unlock (test.types.p11.Unlock *parameters*)

What you see are very prominent methods such as lookup(), findResources(), getProperty(), or getContent(), but also lock()/unlock() and checkOut()/checkIn() as well as mass calls (operating on multiple resources in one Web service call) to reduce the network traffic by reducing the number of calls. Mass calls are a very important topic, because a Web services API must be designed with consideration of the fact that each call has an overhead that depends on the network latency –in complete contrast to a Java API, for example, where the single method call does not matter much and the time spent in the method call is more important for the performance of the system. With Web services, the network latency and SOAP overhead becomes very important – so reducing the number of calls is one goal in the design of Web service APIs. We have taken that into account by providing mass calls for the most important methods such as lookupMass() and getPropertyMass(), but we have also added a data structure named FetchGroup, which you can use to control the actual data fields of the resources returned with the Web service calls. Using FetchGroup, you can control whether or not optional data such as properties (and even which ones), content metadata, or certain access URLs should be returned.

You can find below an overview of the Web service methods currently supported for the index management service:



Config1Port

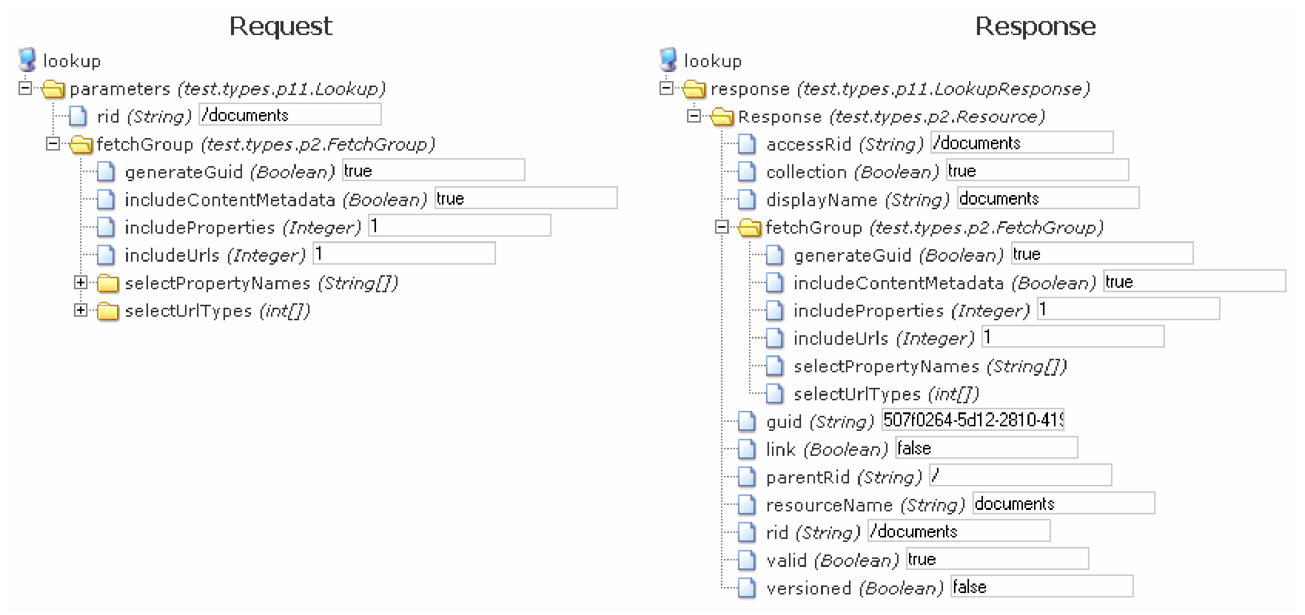
Operations
attachDataSource (test.types.p10.AttachDataSource *parameters*)
clear (test.types.p10.Clear *parameters*)
createIndex (test.types.p10.CreateIndex *parameters*)
deindexResource (test.types.p10.DeindexResource *parameters*)
deindexResources (test.types.p10.DeindexResources *parameters*)
deleteIndex (test.types.p10.DeleteIndex *parameters*)
deltaReindex (test.types.p10.DeltaReindex *parameters*)
detachDataSource (test.types.p10.DetachDataSource *parameters*)
flushQueue (test.types.p10.FlushQueue *parameters*)
getAllIndexIds (test.types.p10.GetAllIndexIds *parameters*)
getBuildInfo (test.types.p10.GetBuildInfo *parameters*)
getGlobalCfg (test.types.p10.GetGlobalCfg *parameters*)
getGlobalPerf (test.types.p10.GetGlobalPerf *parameters*)
getResourceContentAsHtml (test.types.p10.GetResourceContentAsHtml *parameters*)
getSessionPerf (test.types.p10.GetSessionPerf *parameters*)
indexResource (test.types.p10.IndexResource *parameters*)
indexResources (test.types.p10.IndexResources *parameters*)
ping (test.types.p10.Ping *parameters*)
reindex (test.types.p10.Reindex *parameters*)
resetGlobalPerf (test.types.p10.ResetGlobalPerf *parameters*)
resetSessionPerf (test.types.p10.ResetSessionPerf *parameters*)
searchFirstChunkInFolders (test.types.p10.SearchFirstChunkInFolders *parameters*)
searchFirstChunkInIndexes (test.types.p10.SearchFirstChunkInIndexes *parameters*)
searchNextChunk (test.types.p10.SearchNextChunk *parameters*)
setGlobalCfg (test.types.p10.SetGlobalCfg *parameters*)

Here we have added the means to create or query indexes and, most importantly, to search for resources within KM. All calls potentially returning a lot of resources can be handled in chunks, that is, you can specify that you want to receive the first 10 results, then, with the next call, the next 10 results, and so on. We have added these means because data volume is significant in a remote scenario with XML marshalling and unmarshalling (with regard to processing time, memory consumption, and network capacity for the verbose XML data) and something like Java iterators are not available in Web services, of course.

Sample

Its is not easy to present some standard samples, because a Web service consumer uses their own infrastructure to generate a proxy to our Web services and this proxy depends on the infrastructure, platform, and language chosen. But that is also the great thing about Web services – no one is bound to a specific environment – it is an open standard!

The tool we at SAP use to test Web services deployed on the SAP J2EE Engine is the SAP Web Service Navigator, which is part of ESI. The screenshots above are from this tool. You can pick a method from the list in the UI, parameterize it, and execute it and receive the return structure in a tree view. We do this below for the most prominent method, lookup():


```
Request
lookup
  parameters (test.types.p11.Lookup)
    rid (String) /documents
    fetchGroup (test.types.p2.FetchGroup)
      generateGuid (Boolean) true
      includeContentMetadata (Boolean) true
      includeProperties (Integer) 1
      includeUrls (Integer) 1
      selectPropertyNames (String[])
      selectUrlTypes (int[])
```

```
Response
lookup
  response (test.types.p11.LookupResponse)
    Response (test.types.p2.Resource)
      accessRid (String) /documents
      collection (Boolean) true
      displayName (String) documents
      fetchGroup (test.types.p2.FetchGroup)
        generateGuid (Boolean) true
        includeContentMetadata (Boolean) true
        includeProperties (Integer) 1
        includeUrls (Integer) 1
        selectPropertyNames (String[])
        selectUrlTypes (int[])
      guid (String) 507f0264-5d12-2810-41
      link (Boolean) false
      parentRid (String) /
      resourceName (String) documents
      rid (String) /documents
      valid (Boolean) true
      versioned (Boolean) false
```

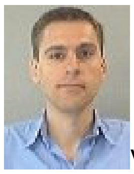
You can see here on the left-hand side that we have requested a resource with the RID */documents*. We want a GUID for it and content metadata, if available. We do not want any properties or URLs to be attached to the resource (constant 1). On the right-hand side, you can see the result: The data transfer object for the resource with default fields and the GUID as requested in the FetchGroup along with the lookup() call.

From a programming perspective, this request depends only on your Web service infrastructure and the generated proxy or generic API for Web services in your Web service infrastructure. But the logic always remains the same: Create and fill out your parameter data structures, make the call, and evaluate the returned data structure(s). In the event of errors, you receive SOAP faults that are based on our central exception handling for Web services.

Statistic and performance data is stored internally and can be retrieved using Web service methods generated in addition to all our Web services, so you can also use Web Services for remote evaluation of the latest exception, number of calls, and processing time.

We also provide JavaDoc-like documentation for all Web services, describing the methods, complex types, fields, exceptions, and even constants for fields for complex types, like the two you have seen in the sample for the lookup() call for properties and URLs. In doing this, we hope to have laid a good foundation for the consumers of our Web services to easily implement remote KM scenarios on whatever platform they chose.

Author Bio


Vedran Lerenc is a member of the Knowledge Management group and the team that designed and developed the new Web services on top of Knowledge Management.