Adding a new Knowledge Base in ServiceCenter 6.2

How to add a new Knowledge Base to Knowledge Management ServiceCenter 6.2

HP® Management Software **Service Management**

Introduction 2

Prerequisites 2

Creating the new Knowledge Base 2
  Create a new kmknowledgebase record 2
    Knowledge Base Scripts 3

Enabling advanced search on the new Knowledge Base 6
  Add new fields to the kmquery dbdict 6
  Modify kmknowledgebase.advsearch.g format 7

Modify the KMSearch ScriptLibrary record 9

Create a Read-Only Viewing Format 12

Assign Format name in kmquery.linkrequest Process Record 12

Add necessary links in kmquery link record for Advanced Search 13

For more information 15

Introduction

In ServiceCenter 6.2 Knowledge Management it is not possible out-of-box to add new ServiceCenter tables as Knowledge Bases. This document will discuss how to add a new ServiceCenter table as a Knowledge Base, including instructions on how to temporarily enable this functionality. As an example, we will add the knownerror table to Knowledge Management as a Knowledge Base.

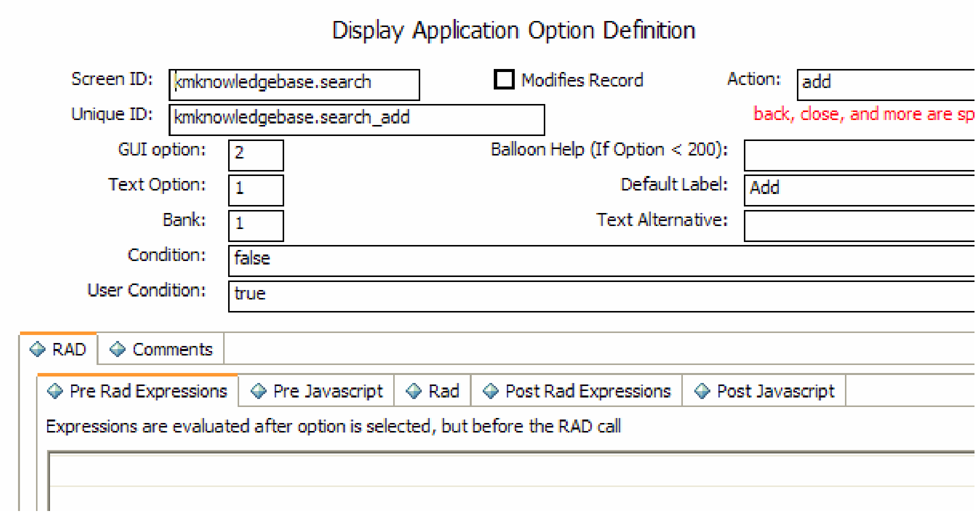
Prerequisites

- ServiceCenter 6.2 with Knowledge Management
- Search Engine installed and configured
- SysAdmin access to ServiceCenter 6.2 via the Windows Client

Creating the new Knowledge Base

Create a new kmknowledgebase record

Preparation: Go to **Utilities – Tools – Display Options** and select the record with Screen ID **kmknowledgebase.search** and Unique ID **kmknowledgebase.search_add**. Temporarily change the User Condition to **true**.



Display Application Option Definition

Screen ID: kmknowledgebase.search

□ Modifies Record

Action: add

back, close, and more are sp

Unique ID: kmknowledgebase.search_add

GUI option: 2

Balloon Help (If Option < 200):

Text Option: 1

Default Label: Add

Bank: 1

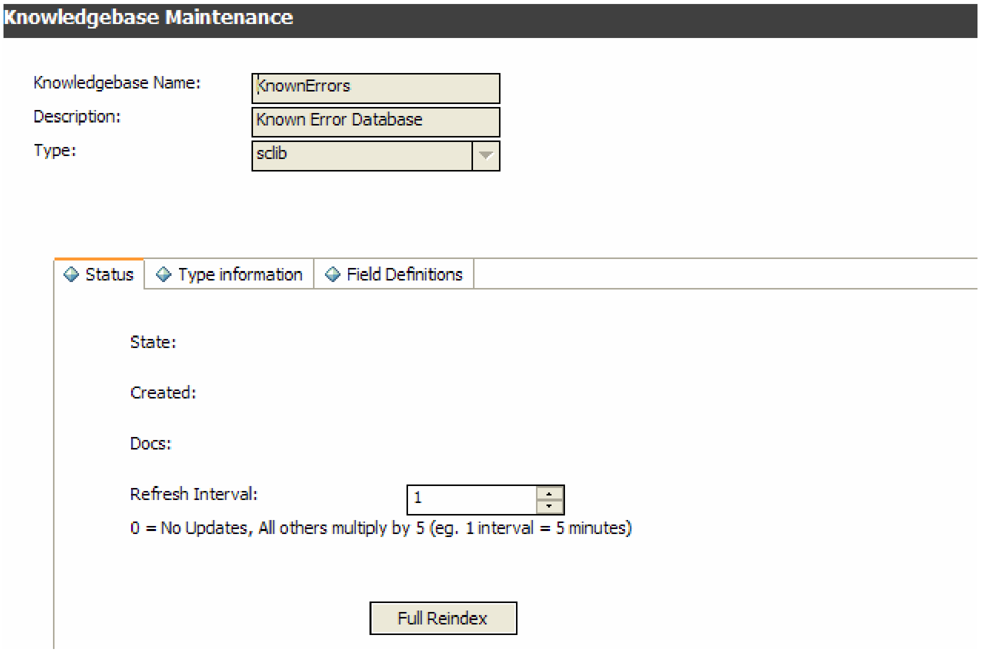
Text Alternative:

Condition: false

User Condition: true

Expressions are evaluated after option is selected, but before the RAD call

1. From the main Menu, go to **Knowledge Management**
2. Click on **Manage Knowledgebases**
3. Fill in the new Knowledge Base name and select **sclib** for the type.
4. Click **Add**.
5. Fill in the **Refresh Interval** on the Status tab (1 in this example equals 5 Minutes).
6. Click **Save**.



Knowledgebase Maintenance

Knowledgebase Name: KnownErrors

Description: Known Error Database

Type: sclib

State:

Created:

Docs:

Refresh Interval: 1

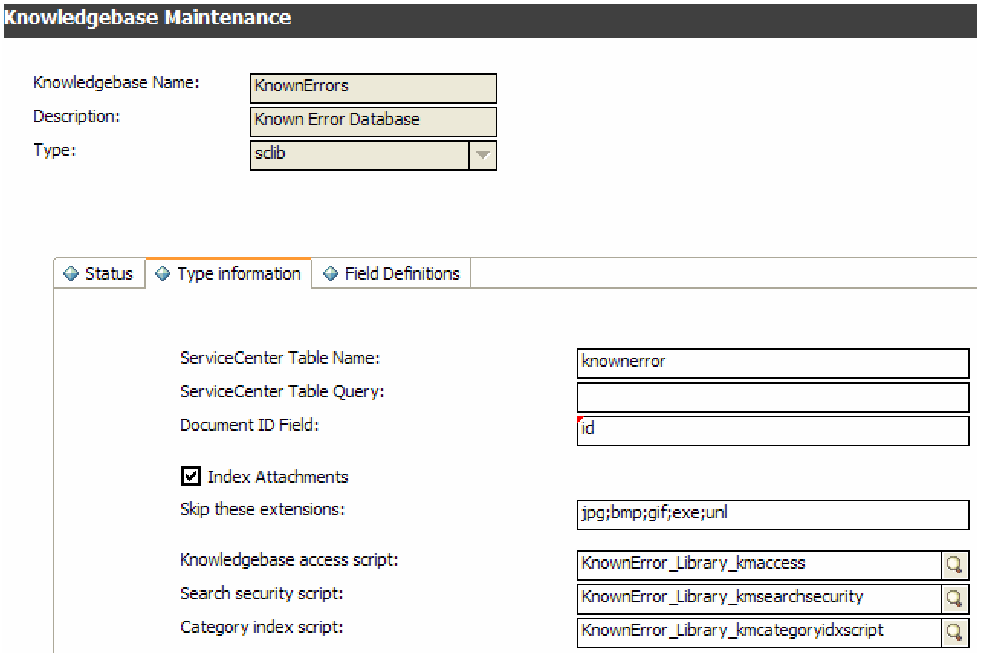
0 = No Updates, All others multiply by 5 (eg. 1 interval = 5 minutes)

7. Fill in all necessary values on the **Type information**.

   The ServiceCenter table name is the name of the dbdict record for the table that should be indexed. The Document ID field needs to be filled with that dbdict's Unique Key. The ServiceCenter Table Query is optional in case not all records in that table are supposed to be indexed.

   Note: We recommend to index attachments, but exclude pictures, executables and unload files.

   The scripts are mandatory and can be copied from existing Knowledge Base scripts. See the section on Knowledge Base Scripts below for more information on what information these scripts should contain.



Knowledgebase Maintenance

Knowledgebase Name: KnownErrors

Description: Known Error Database

Type: sclib

ServiceCenter Table Name: knownerror

ServiceCenter Table Query:

Document ID Field: id

☑ Index Attachments

Skip these extensions: jpg;bmp;gif;exe;unl

Knowledgebase access script: KnownError_Library_kmaccess

Search security script: KnownError_Library_kmsearchsecurity

Category index script: KnownError_Library_kmcategoryidxscript

Knowledge Base Scripts

Kmaccess Script – KnownError_Library_kmaccess:

```
// This script determines user access to a knowledgebase
```

```
function checkAccess(user, record)
{
  // user is the name of the user.
  // record is the kmkbase record for a collection,
  // it contains the collection name and the file being indexed.
  // This function must return True or False, depending on
  // if the user was validated to search this collection or not.
  // The validation method should check that the user can see the
  // filename being indexed at it's base level, For example,
  // a user can search probsummary if they have incident management
rights
  //
  // Since a single file can have multiple indexes, you can also
  // validate the user against the knowledgebase name.  Remember,
  // this function must return True or False.  How you determine
  // which one to return is up to you.

  // check if user has access to record.sclibtablename
  // if yes,
  // For Incident Management (interactions file), look for browse
  //   in the Incident Profile in the operator record
  // But for the ESS user, no access
  if (system.functions.nullsub(vars.$G_ess,false))
        return false;
  else
      return
system.functions.nullsub(vars.$G_rc_environment.browse,false);
}
```

Adjust the $G_xx_environment variable name to the Profile variable used for this table. To find the correct variable name, go to Database Manager and bring up the record for this table in the Object table. The Profile Variable is listed on the Object Info tab. Replace all dots in the variable name by underscores for correct translation from Service Manager to JavaScript syntax.

Search Security Script - KnownError_Library_kmsearchsecurity

```
// This script returns a string containing a search query
// that filters the search based on the user's rights and
// permissions.


function getSecurityInfo(user, record)
{
  return "";
}
```

No adjustments needed, unless detailed Search Security is needed. For more information on Search Security, refer to the White Paper with the title *Using Mandanten Protection in the ServiceCenter® Knowledge Management Module.*

Category Index Script - KnownError_Library_kmcategoryidxscript

```
function getCategoryStr(record)
{
  var categoryStr = null;
  //
  // check the filename

  if (system.functions.filename(record) == "knownerror")
  {
    if (record.category != null)
    {
      categoryStr = record.category;
      if (record.subcategory != null)
      {
        categoryStr += ":"+ record.subcategory;
        if (record.product_type != null)
        {
          categoryStr +=  ":"+ record.product_type;
          if (record.problem_type != null)
          {
            categoryStr +=  ":"+ record.problem_type;
            //categoryStr +=  ";"+ categoryStr;
          }
        }
      }
    }
  }
  else
    categoryStr = "";

  return categoryStr;
}
```

In this script change the system.functions.filename(record) value to the name of the dbdict that belongs to your table (here “knownerror”). If the table you are including in the knowledge base uses different categorization, change the names of the category fields here.

8. Fill in the Field definition tab:

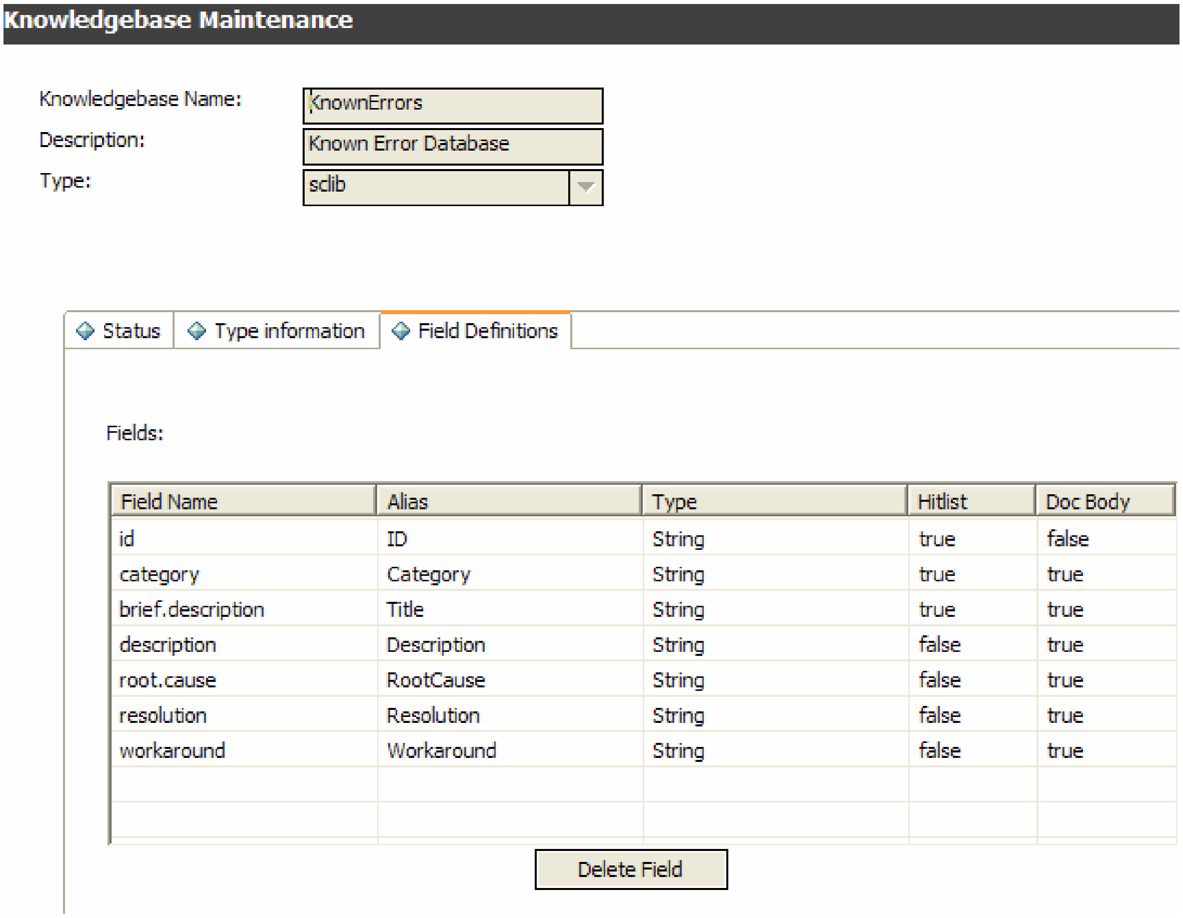
   Add all fields here that need to be in the Hitlist or need to be indexed (Doc Body).

   The Field column needs to contain the field name as defined in the dbdict.

   The Alias column determines what the field will be called within Knowledge Management.

   Fields that should show in the Hitlist have to have a true value in that column (otherwise false), fields that are supposed to be included in the index need to have a true value in the Doc Body column.

   Note: Fields that should be shown in the Knowledge Article without being indexed or in the Hitlist do not have to be added here. These fields will be added to the View and the Forms for the Knowledge Document type.


Knowledgebase Maintenance

Knowledgebase Name:	KnownErrors
Description:	Known Error Database
Type:	sclib

Fields:

Field Name	Alias	Type	Hitlist	Doc Body
id	ID	String	true	false
category	Category	String	true	true
brief.description	Title	String	true	true
description	Description	String	false	true
root.cause	RootCause	String	false	true
resolution	Resolution	String	false	true
workaround	Workaround	String	false	true

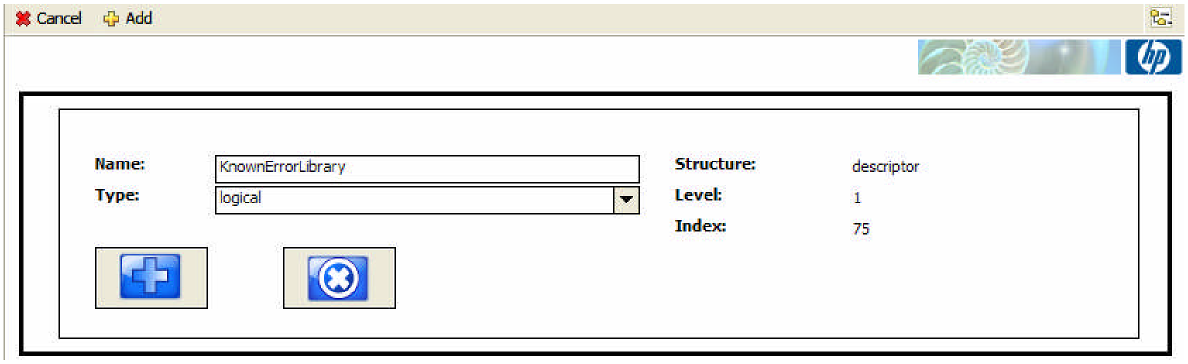
9. Click **Save** to save the record
10. On the Status tab, click on **Full-Reindex**
11. Click **OK** to save and exit.
12. Go to **Utilities – Tools – Display Options** and select the record with Screen ID **kmknowledgebase.search** and Unique ID **kmknowledgebase.search_add**. Remove the User Condition of **true** and click **OK** to save and exit..

Enabling advanced search on the new Knowledge Base

Add new fields to the kmquery dbdict

Follow the naming convention *libraryname.fieldname* (i.e. knownerrorlib.id) and add new fields to the kmquery dbdict to enable advanced searching on specific fields in the new knowledgebase.

1. Go to Toolkit – DBDict Utility and search for the kmquery dbdict record.
2. Click on the descriptor field and click on New.
3. First add a new Boolean field for your library to be able to include or exclude it in searches.



Name: KnownErrorLibrary

Type: logical

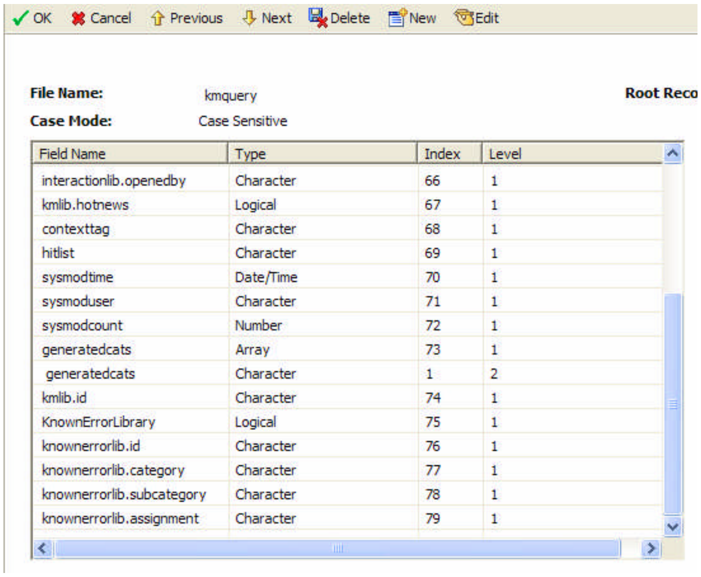
Structure: descriptor

Level: 1

Index: 75

4. Enter the following new character type fields:
   - knownerrorlib.id
   - knownerrorlib.category
   - knownerrorlib.subcategory
   - knownerrorlib.assignment
5. Click **Add** for each of the fields.

   Note: These fieldnames are used in the format input and are mapped into the query in the KMSearch script library



File Name: kmquery

Root Reco

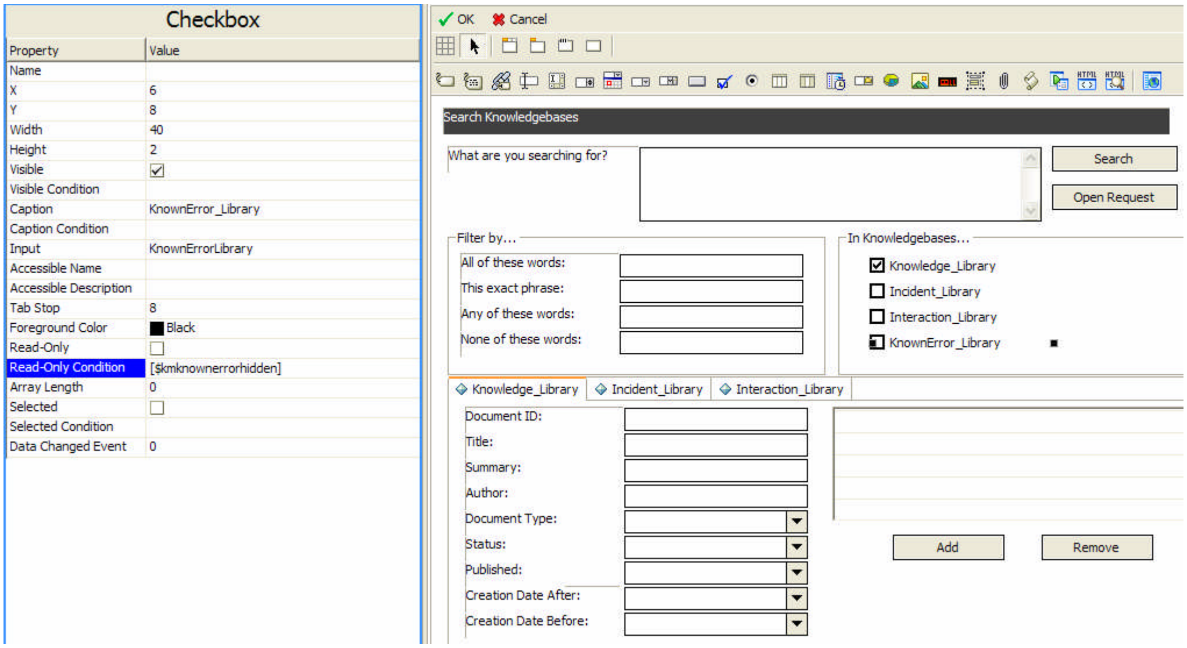
Case Mode: Case Sensitive

Field Name	Type	Index	Level
interactionlib.openedby	Character	66	1
kmlib.hotnews	Logical	67	1
contexttag	Character	68	1
hitlist	Character	69	1
sysmodtime	Date/Time	70	1
sysmoduser	Character	71	1
sysmodcount	Number	72	1
generatedcats	Array	73	1
generatedcats	Character	1	2
kmlib.id	Character	74	1
KnownErrorLibrary	Logical	75	1
knownerrorlib.id	Character	76	1
knownerrorlib.category	Character	77	1
knownerrorlib.subcategory	Character	78	1
knownerrorlib.assignment	Character	79	1

6. Click **OK** to save and exit.

Modify kmknowledgebase.advsearch.g format

1. Go to **Toolkit – Forms Designer** and select the **kmknowledgebase.advsearch.g** form
2. Click on **Design** to go into Design Mode
3. Copy one of the existing knowledgebase check boxes and rename values to match your new kb

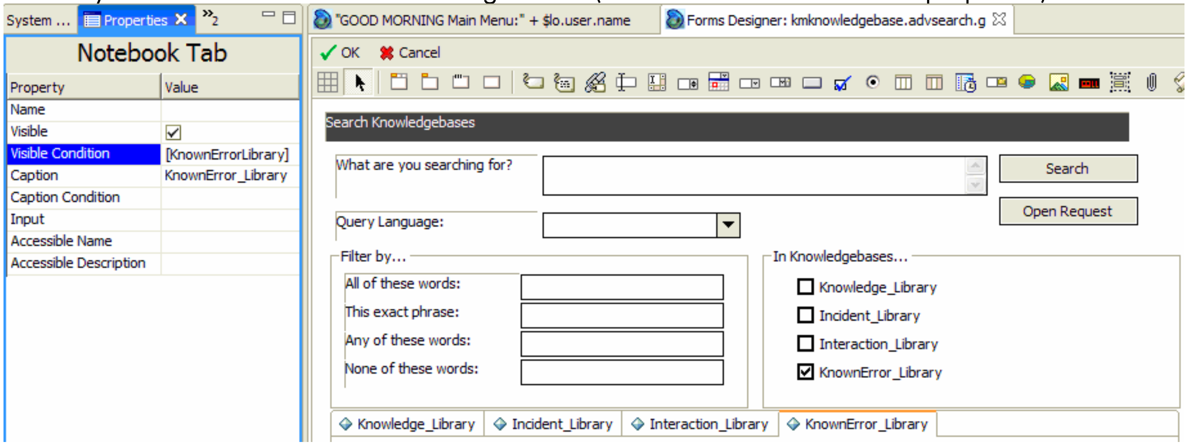


Checkbox

Property	Value
Name	
X	6
Y	8
Width	40
Height	2
Visible	☑
Visible Condition	
Caption	KnownError_Library
Caption Condition	
Input	KnownErrorLibrary
Accessible Name	
Accessible Description	
Tab Stop	8
Foreground Color	■ Black
Read-Only	□
Read-Only Condition	[$kmknownerrorhidden]
Array Length	0
Selected	□
Selected Condition	
Data Changed Event	0

In Knowledgebases...
☑ Knowledge_Library
□ Incident_Library
□ Interaction_Library
☑ KnownError_Library

4. Create a new tab for the new Knowledge Library. Make the new tab visible only when the new Library is checked in the list of knowledge bases (see below for the notebook properties).

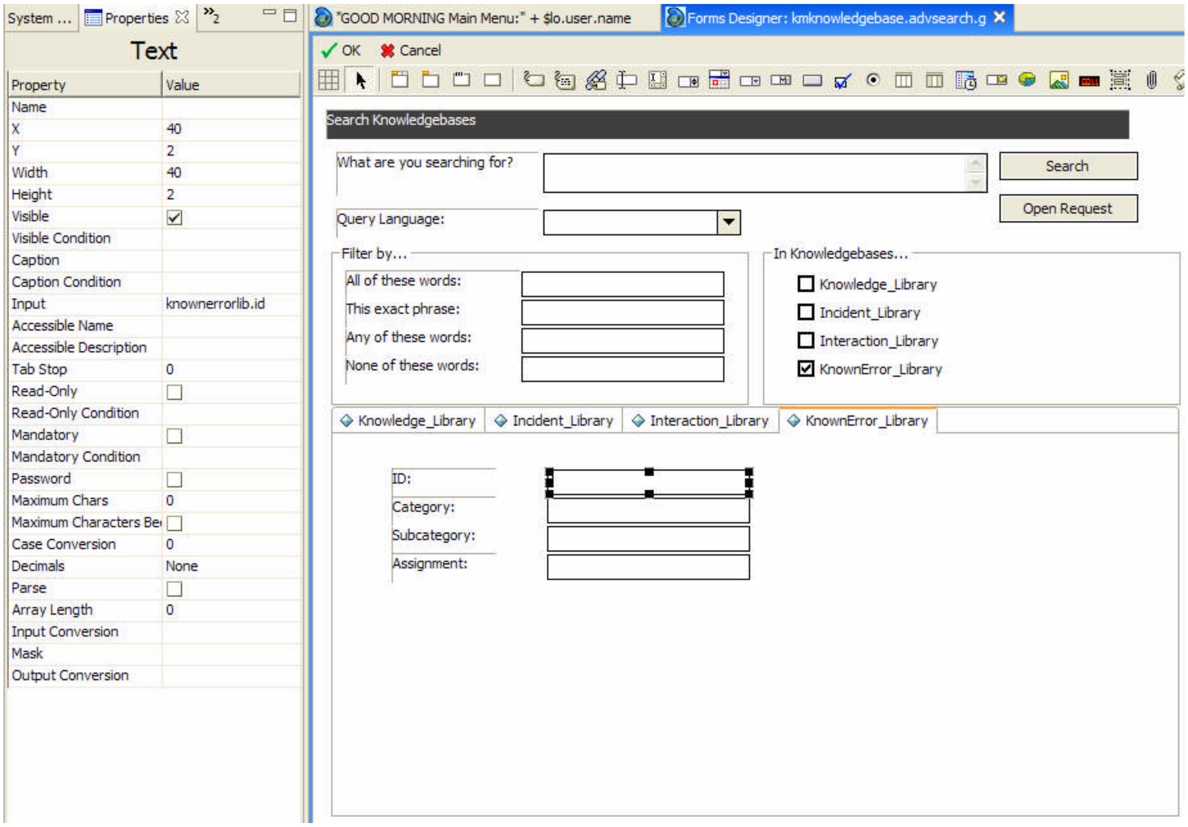


Notebook Tab

Property	Value
Name	
Visible	☑
Visible Condition	[KnownErrorLibrary]
Caption	KnownError_Library
Caption Condition	
Input	
Accessible Name	
Accessible Description	

In Knowledgebases...
□ Knowledge_Library
□ Incident_Library
□ Interaction_Library
☑ KnownError_Library

5. Add all fields that were added to the kmquery dbdict to the new tab as shown below.



Text

Property	Value
Name	
X	40
Y	2
Width	40
Height	2
Visible	☑
Visible Condition	
Caption	
Caption Condition	
Input	knownerrorlib.id
Accessible Name	
Accessible Description	
Tab Stop	0
Read-Only	□
Read-Only Condition	
Mandatory	□
Mandatory Condition	
Password	□
Maximum Chars	0
Maximum Characters Be	□
Case Conversion	0
Decimals	None
Parse	□
Array Length	0
Input Conversion	
Mask	
Output Conversion	

6. **Note**: For easier adding the new fields, go to the **System Navigation – System Definition – Tables – kmquery – Fields** and drag and drop the new fields onto the tab for the new library.
7. Click **OK** twice to save and exit.

Modify the KMSearch ScriptLibrary record

1. Go to the ScriptLibrary by entering sl in the command line and hit enter
2. Select the record by the name of KMSearch
3. Search for the `function getAvailableKnowledgeBases` (Ctrl + F will bring up a search window) and add an if – else section corresponding to your new knowledgebase as shown below in bold font:

```
function getAvailableKnowledgeBases(fKMQuery)
{

   var strKBList =
system.library.KMSearchQuery.getvalidKBs(system.functions.operator());
   if(strKBList.indexOf("Knowledge_Library") >= 0)
   {
      vars.$kmknowledgehidden = false;
      fKMQuery.KnowledgeLibrary = true;
   }
   else
   {
      vars.$kmknowledgehidden = true;
      fKMQuery.KnowledgeLibrary = false;
   }

   if(strKBList.indexOf("Incident_Library") >= 0)
   {
      vars.$kmincidenthidden = false;
      fKMQuery.IncidentLibrary = true;
   }
   else
   {
      vars.$kmincidenthidden = true;
      fKMQuery.IncidentLibrary = false;
   }


   if(strKBList.indexOf("Interaction_Library") >= 0)
   {
      vars.$kminteractionhidden = false;
      fKMQuery.InteractionLibrary = true;
   }
   else
   {
      vars.$kminteractionhidden = true;
      fKMQuery.InteractionLibrary = false;
   }
   // In the double quotes enter the name of the Knowledge Base exactly
   // as defined in Manage Knownledge Bases.

   if(strKBList.indexOf("KnownErrors") >= 0)
   {
      vars.$kmknownerrorhidden = false;
      fKMQuery.KnownErrorLibrary = true;
   }
   else
   {
      vars.$kmknownerrorhidden = true;
      fKMQuery.KnownErrorLibrary = false;
   }
}
```

4. Search for and edit `function getSelectedCollections` and add an if section for your new knowledgebase as shown in bold font below:

```
function getSelectedCollections(fKMQuery)
{
   var colls = new Datum();
   colls.setType(8);
   if(fKMQuery.KnowledgeLibrary)
      colls.push("Knowledge_Library");
   if(fKMQuery.IncidentLibrary)
      colls.push("Incident_Library");
   if(fKMQuery.InteractionLibrary)
      colls.push("Interaction_Library");
   if(fKMQuery.KnownErrorLibrary)
      colls.push("KnownErrors");
   return colls;
}
```

5. Search and edit `function getSelectedCollectionsString` to add an if section corresponding to your new knowledgebase as shown in bold font below:

```
function getSelectedCollectionsString(fKMQuery)
{
   var colls = "";
   if(fKMQuery.KnowledgeLibrary)
      colls +="Knowledge_Library";
   if(fKMQuery.IncidentLibrary)
      colls += ((colls !="" ? ";":"") + "Incident_Library");
   if(fKMQuery.InteractionLibrary)
      colls += ((colls !="" ? ";":"") + "Interaction_Library");
   if(fKMQuery.KnownErrorLibrary)
```

```
      colls += ((colls !="" ? ";":"") + "KnownErrors");
   return colls;
}
```

6. Add a new function named **process<LibraryNameHere>LibCriteria** to handle building the proper query syntax. This is easiest done by copying and pasting an existing function and adjusting it to the requirements for the new Knowledge library. An example below:

```
function processKnownErrorLibCriteria(fKMQuery)
{
   var strQuery = "";
   if(fKMQuery.knownerrorlib_id != null)
      strQuery += " <AND> (id <CONTAINS> " + fKMQuery.knownerrorlib_id +
" <AND> _style <CONTAINS> KnownErrors)";
   if(fKMQuery.knownerrorlib_category != null)
      strQuery += " <AND> (category <CONTAINS> " +
fKMQuery.knownerrorlib_category + " <AND> _style <CONTAINS>
KnownErrors)";
   if(fKMQuery.knownerrorlib_subcategory != null)
      strQuery += " <AND> (subcategory <CONTAINS> " +
fKMQuery.knownerrorlib_subcategory + " <AND> _style <CONTAINS>
KnownErrors)";
   if(fKMQuery.knownerrorlib_assignment != null)
      strQuery += " <AND> (assignment <CONTAINS> " +
fKMQuery.knownerrorlib_assignment + " <AND> _style <CONTAINS>
KnownErrors)";
   return strQuery;
}
```

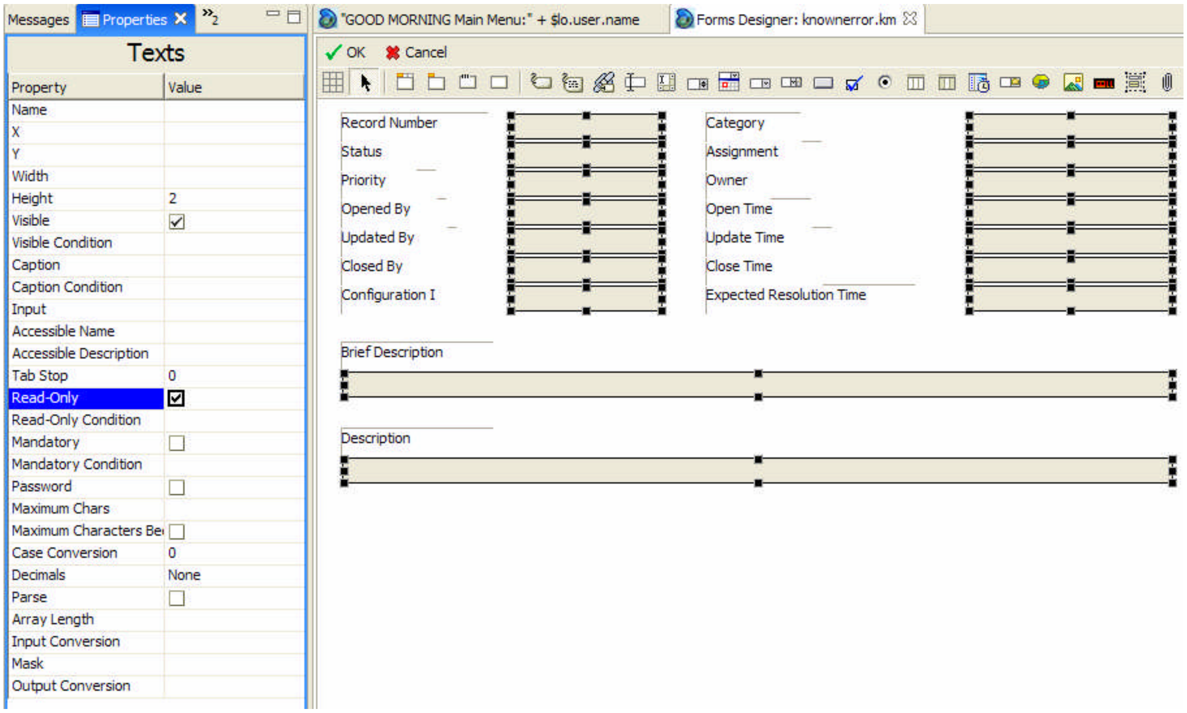
7. The new function defined above is then conditionally called from the **buildQueryString** function as shown in bold font below:

```
function buildQueryString( fKMQuery )
{
[…]
   //process exact phrase
   if( fKMQuery.exactphrase != null && fKMQuery.exactphrase.length > 0 )

   {
      strQuery += " <AND> (\"" + fKMQuery.exactphrase + "\")";
      //strQuery += " \"" + fKMQuery.exactphrase + "\"";
   }

   if(fKMQuery.KnowledgeLibrary)
      strQuery += system.library.KMSearch.processKMLibCriteria(fKMQuery);

   if(fKMQuery.IncidentLibrary && fKMQuery.kmlib_hotnews != true)
      strQuery +=
system.library.KMSearch.processIncidentLibCriteria(fKMQuery);

   if(fKMQuery.InteractionLibrary && fKMQuery.kmlib_hotnews != true)
      strQuery +=
system.library.KMSearch.processInteractionLibCriteria(fKMQuery);

   if(fKMQuery.KnownErrorLibrary && fKMQuery.kmlib_hotnews != true)
      strQuery +=
system.library.KMSearch.processKnownErrorLibCriteria(fKMQuery);

   strQuery +=
system.library.KMSearchQuery.getsearchSecurity(system.functions.operator(
), this.getSelectedCollectionsString(fKMQuery))

   fKMQuery.submitstring = strQuery.indexOf(" <AND> ")==0 ?
strQuery.substring(6, strQuery.length) : strQuery;
   //print("Submitted Query: "+fKMQuery.submitstring);
   return fKMQuery;
}
```

Create a Read-Only Viewing Format

1. To create a read-only viewing format specifically for use as a knowledge candidate go to Forms Designer and search for an existing form on the table used for the new knowledge base that contains all fields required.
2. Copy the form that meets the requirement, here the knownerror.reminder form, to **knownerror.km** and change the read-only condition on all fields to **true**.
3. Click **OK** twice to save and exit.

Note: Other sample forms for this functionality are: IM.template.km, SD.view.interaction.km.



Texts

Property	Value
Name	
X	
Y	
Width	
Height	2
Visible	☑
Visible Condition	
Caption	
Caption Condition	
Input	
Accessible Name	
Accessible Description	
Tab Stop	0
Read-Only	☑
Read-Only Condition	
Mandatory	□
Mandatory Condition	
Password	□
Maximum Chars	
Maximum Characters Be	□
Case Conversion	0
Decimals	None
Parse	□
Array Length	
Input Conversion	
Mask	
Output Conversion	

Assign Format name in kmquery.linkrequest Process Record

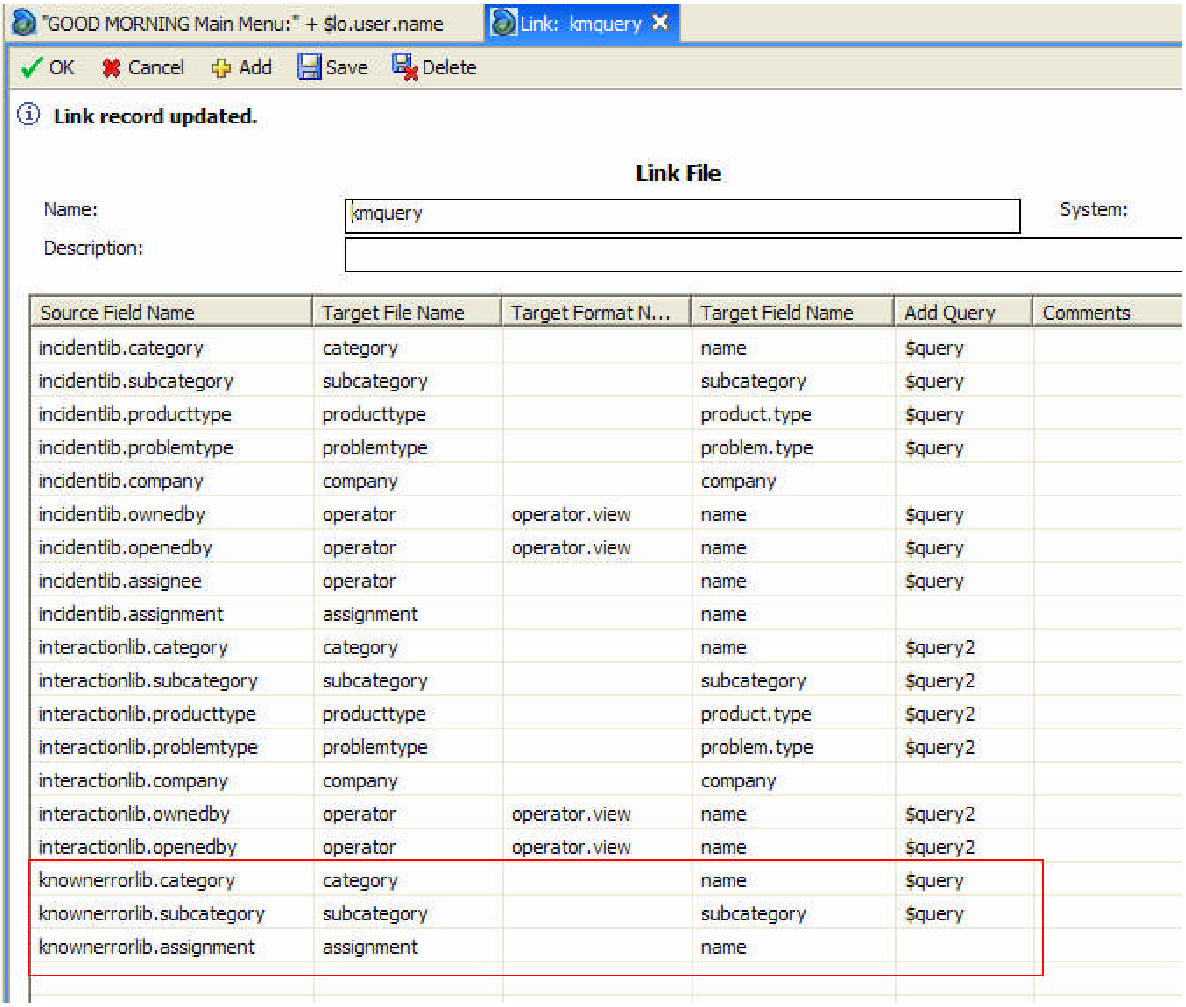
1. The format name added above must be assigned in the Initial Javascript tab of the kmquery.linkrequest Process record.
2. To do so, go to **Utilities – Tools – Document Engine – Processes** and select the **kmquery.linkrequest** Process.
3. Add a corresponding else-if section in the Initial JavaScript section of the Process to assign the format name to be used for the new knowledgebase as shown in bold font below:

```
var strId = system.library.KMUtils.removetoken(vars.$L_linkquery, "\"");
strId = strId.split("=");

vars.$L_doctoview = new SCFile(vars.$L_linktable);
var rc = vars.$L_doctoview.doSelect(vars.$L_linkquery);

vars.$L_modetouse = "browse";

if( vars.$L_linktable == "kmdocument" )
{
   if (vars.$L_doctoview.doctype == "external")
      vars.$L_formattouse="kmdocument.viewexternalwithview";
   else
```

```
      vars.$L_formattouse="kmdocument.viewwithview";
   vars.$firepreview=true;
}
else if( vars.$L_linktable == "probsummary" )
{
   //vars.$L_modetouse = "KMbrowse";  //see im.view.init process record
   vars.$L_formattouse="IM.template.km";
}
else if(vars.$L_linktable == "incidents")
{
   //vars.$L_modetouse = "KMbrowse";  //see ess.determine.EdtFmt process
record
   vars.$L_formattouse="SD.view.interaction.km";
}
else if(vars.$L_linktable == "knownerror")
{
   //vars.$L_modetouse = "KMbrowse";  //see ess.determine.EdtFmt process
record
   vars.$L_formattouse="knownerror.km";
}

system.library.KMUsageStats.incrementViewCount(strId[1],
vars.$L_linktable);
system.library.KMUsageHistory.viewed(strId[1], vars.$L_linktable);

vars.$L_kmviewedid = strId[1];
```

Add necessary links in kmquery link record for Advanced Search

1. To be able to use the Fill buttons on fields in the advanced search in Knowledge Management the kmquery link has to be updated. To do so go to Utilities – Tools – Links and search for the link by the name of kmquery.
2. Add the fields to the link record that need to be filled, such as knownerrorlib.category, knownerrorlib.subcategory and knownerrorlib.assignment.



Link record updated.

Link File

Name: kmquery

System:

Description:

Source Field Name	Target File Name	Target Format N...	Target Field Name	Add Query	Comments
incidentlib.category	category		name	$query	
incidentlib.subcategory	subcategory		subcategory	$query	
incidentlib.producttype	producttype		product.type	$query	
incidentlib.problemtype	problemtype		problem.type	$query	
incidentlib.company	company		company		
incidentlib.ownedby	operator	operator.view	name	$query	
incidentlib.openedby	operator	operator.view	name	$query	
incidentlib.assignee	operator		name	$query	
incidentlib.assignment	assignment		name		
interactionlib.category	category		name	$query2	
interactionlib.subcategory	subcategory		subcategory	$query2	
interactionlib.producttype	producttype		product.type	$query2	
interactionlib.problemtype	problemtype		problem.type	$query2	
interactionlib.company	company		company		
interactionlib.ownedby	operator	operator.view	name	$query2	
interactionlib.openedby	operator	operator.view	name	$query2	
knownerrorlib.category	category		name	$query	
knownerrorlib.subcategory	subcategory		subcategory	$query	
knownerrorlib.assignment	assignment		name		

For more information

Please visit the HP Management Software support Web site at:

http://www.hp.com/managementsoftware/support

This Web site provides contact information and details about the products, services, and support that HP Management Software offers.

HP Management Software online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued customer, you can benefit by being able to:

- Search for knowledge documents of interest
- Submit and track progress on support cases
- Submit enhancement requests online
- Download software patches
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Note: Most of the support areas require that you register as an HP Passport user and sign in. Many also require an active support contract.

To find more information about support access levels, go to the following URL:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to the following URL:

http://www.managementsoftware.hp.com/passport-registration.html

01/2008

invent