Document number	Revision
DOCU12421	12

Advanced schema and customization in Highstage

Advanced schema and customization in Highstage

1. Introduction

This article will help you get started with changing behavior or add custom functionality to Highstage through schema configuration.

Please note that potential changes to the custom XML schemas can cause errors resulting in potential downtime of Highstage. Thus, consider using a designated test server during development, and implement modifications on the production server out of business hours.

2. Scope

This article serves both Highstage clients and personnel.

3. Add a new database column for items

Here, you will be introduced to the procedure of adding a new database column to an exiting type/table.

3.1. Add column to type

In `custom.schema.xml` add the column definition to a specific type, for example, ***doc*** type:

```
<type name="doc">
    <column name="country" description="" sqltype="varchar" sqlsize="10" sqlnull="0" sqldefault="('')" />
</type>
```

> Observe that ***part*** type inherits from ***doc*** type, so the new column will also be available in part type.

4. Extend the length of an existing database column

In some situations, it may be required to extend column length.

This may be done directly in the database by using SQL management studio, or by the schema. Changing column length or type may require unchecking ***Prevent saving changes that require table re-creation***.

This can be disabled by navigating to Tools>Option>Designers:

The following describes how to extend the Author column by the schema.

To find the name of the physical database column, run as SuperUser *(or higher)* and click on the grid or form title:

The physical column appears as ***editby***:

Change `custom.schema.xml` to extend the column length, for example to 512 bytes:

```
<type name="ts_item">
    <column name="editby" sqlsize="512" />
</type>
```

The length of a physical database column will be extended to match the schema definition by running Apply schema web page.

> You can apply the schema changes by navigating to SYSTEM > SCHEMA > APPLY from the Highstage side navigation menu with *SuperUser*/AdminRead/AdminWrite user level enabled.

This approach using schema can only be used to extend column length. It cannot be used to shorten the length or to change the type of a physical database column. To shorten the length or change SQL type of a database column the SQL management studio must be used.

5. <type name="_file" and <refinery name="file>" elements

The _file type controls how files are handled. The element is inherited by other types like doc.

Besides defining columns and code required for handling files it also specifies refine behavior, the `<refinery name="file">` element does that. Currently, both new Aspose based refinery and old MS-Office based refinery is supported, in the future when MS-Office refinery is depreciated the XML will be more simple. This documentation, however, will only describe the XML required for Aspose based refinery:

```xml
    <type name="_file">
      <refinery name="file" refine="1" freeprimaryfilename="1" confidentialitylevel="" acceptchanges="4"
  renditionstep="2" renditionexcludemarkup="4" >
        <!-- refine: 0:disable, 1:Primary file, 2:Matching, 3:All in folder -->
        <ext name=".doc" app="ms_word" />
        <ext name=".docx" app="ms_word" />
        <ext name=".docm" app="ms_word" />
        <ext name=".odt" app="ms_word" />
        <ext name=".xls" app="ms_excel" />
        <ext name=".xlsx" app="ms_excel" />
        <ext name=".xlsm" app="ms_excel" />
        <ext name=".ppt" app="ms_powerpoint" />
        <ext name=".pptx" app="ms_powerpoint" />
        <ext name=".pptm" app="ms_powerpoint" />
        <ext name=".mpp" app="ms_project" _delete="1" />
        <ext name=".vsd" app="ms_visio" _delete="1" />
        <ext name=".vsdx" app="ms_visio" _delete="1" />
        <!-- properties attribute may be set at app node (like refinery node) if specific property
  requirements exists for specific applications -->
        <app name="ms_word" rendition="pdf" refine-url="ts/refinery/aspose/refine.exe.aspx" />
        <app name="ms_excel" rendition="pdf" refine-url="ts/refinery/aspose/refine.exe.aspx" />
        <app name="ms_powerpoint" rendition="pdf" refine-url="ts/refinery/aspose/refine.exe.aspx" />
        <app name="ms_project" refine-url="ts/refinery/aspose/refine.exe.aspx" />
        <app name="ms_visio" rendition="pdf" refine-url="ts/refinery/aspose/refine.exe.aspx" />
      </refinery>
    </type>
```

Sample snippet that changes refine of the primary file to all files in folder matching item meaning starting with item id:

```xml
    <type name="_file">
      <refinery name="file" refine="2" />
    </type>
```

6. <feature name="config-manager"> element

The ***feature*** element with the name ***config-manager*** supports various attributes that define how items should be named and how revision management should work. The feature element is related to the type in which it is declared. The following is the default declaration, in this case defining doc type behaviors:

```
<type name="doc">
    <feature name="config-manager"
    bizunitprefixinid="0"
    delimiter1=""
    subtypeinid="1"
    delimiter2=""
    alphaid="0"
    delimiter3="-"
    alphaversion="0"
    alpharevision="1"
    delimiter4=""
    preliminary1=""
    preliminary2=""
    mail-oncreate="1"
    majorRevisions="0"
    minorRevisions="0"
    ClearChangeNoteOnChange="1"
    RefHardLock="0"
    RefHardLockStatusEnforcement="2;3;4"
    review-template=""
    change-undo="0"
    SoftTemplates="1"
    change-write="resource1;manager;teammember;superuser;defaulttrustee"
    cloak-uncloak="0"
    version-numeric-start-value="1"
    />
</type>
```

Name	Values	Description
`alphaid`	0 *(default)* / 1	Configure the serial ID of the item to be consisting of numbers (`alphaid="0"`) or letters (`alphaid="1"`).
`bizunitprefixinid`	0/1	Use the ID of the business unit that created the document. May be used in multi-business unit organizations to clarify origin. Besides setting this value to 1, the parameter *BizUnitPrefix* must be set to a proper string value, the string value should be kept short, for example, max 3 chars.
`delimiter1`	char	Delimiter between *bizunit* and *subtype*.
`subtypeinid`	0/1	Use subtype id in document id. This would normally be the case. However, in some rare cases, there could be a need for the ability to change action subtype without changing the action id.
`delimiter2`	char	Delimiter between subtype and serial number id.
`delimiter2`	0/1	Use alphanumeric instead of numeric serial number id.
`delimiter3`	char	Delimiter between serial number id and version.
`alphaversion`	0/1	Use alphanumeric instead of numeric version.
`alpharevision`	0/1	Use alphanumeric instead of numeric revision.
`delimiter4`	char	Delimiter between version and revision.
`preliminary1`	char	Placed before version until item has been approved.
`preliminary2`	char	Placed after revision until item has been approved.
`mail-oncreate`	0/1 *(default)*	Enable to allow for mails to be sent to appropriate users' when new items of the associated type is created *(requires active email notification triggers to be configured)*.
`majorRevisions`	0/1	Enable major revisions.
`minorRevisions`	0/1	Enable minor revisions *(minor change)*. When making a minor change to an Approved document (if enabled), the document's status changes to **Working**, allowing authors to update its data, references, and files. After the changes are made, the document follows the standard **review and approval process** to complete the initiated *Minor change*. The main version of the document does not change, but a **minor version history** is recorded in separate fields: *Minor Version* and *Minor Revision*. Existing references to- and from the Document remains unchanged as the primary *Version* is retained. **Important**: A minor change must be fully **Approved** before starting any major change to the document.
`ClearChangeNoteOnChange`	0/1	Clear change note when item is changed.

Name	Values	Description
`RefHardLock`	0/1	When you make a change to a non-approved item *(i.e. where the item is in Freeze/Review/Approval)*, locked references will continue referencing the existing version of an item. Updating references must be manually maintained by users.
`RefHardLockStatusEnforcement`	2/3/4	Requires enabling of `RefHardLock` (`RefHardLock="1"`) to be enforced. Specify which specific document statuses *(Freeze, Review, Approval)* from which references may not be automatically updated when making a change to an unapproved document. If a specific status is omitted, the locked reference to a non-approved item with that specific status will be automatically updated as if `RefHardLock` was disabled.
`review-template`	subtype name	Specifies which subtype to be used as review template. The following would set review template to the type MÓM *(Minutes Of Meeting)*: `<type name="doc" <feature name="config-manager" review-template="MOM" /> </type>`
`softtemplates`	0/1	If you change an approved item, the folder will be updated with the contents of the item specified in the 'SoftTemplate' data column. If no version is specified, the latest approved version's contents will be used.
`change-undo`	0/1	Enables the the option for users to undo an immediate *Change* on an item to revert back to the previous version prior to the change. The *UNDO-CHANGE* will delete the current version of the item altogether and promote the previous version of the item to the latest. The UNDO-CHANGE button, when enabled, is available within the Advanced section *(requires a set user level of AdvancedUser)* on any item. Disabled *(default)*: `change-undo="0"` . Enabled: `change-undo="1"` .
`change-write`	List of vector-roles	change-write determines which vector roles that is allowed to perform a change on item. Default is: `resource1` , `manager` , `teammember` , `superuser` *Complete list of vector roles can be found in SYSTEM > DIAGNOSTICS -> VECTOR ROLES*
`cloak-uncloak`	0/1	Inventor MCAD specific feature. Enable button author that makes it possible to hide folder from author.

Name	Values	Description
`min-status-child-enforcement`	1-5 *(default: 2)*	Parameter controlling the enforcement of all references to Documents/Parts/Devices to be updated *(having the same or later status compared to the parent)* before the parent item can be stepped to the same status. This parameter allows for you to specify a select status *(of the parent item)* from where the children are required to have the same *(or later)* status as the parent. For references to be considered for this enforcement, their associated basetype must be assigned to the `min-status-children-scope`. Default: `min-status-child-enforcement="2"` *(parent item can be stepped to Freeze without enforcing all child references to be in Freeze beforehand. For Review/Approval/Approved - child references are required to have, at least, the same status before the parent item be stepped.*
`min-status-children-scope`	List of basetypes	Parameter controlling which basetypes *(default or custom)* of child references to be considered for `min-status-child-enforcement`. Default: `min-status-children-scope="doc;part"`
`version-numeric-start-value`	Number. Default value: 1	Parameter controlling the revision starting value. Must be a numeric value. To change to 0, add following to custom.schema: `<type name="doc"> <feature name="config-manager" version-numeric-start-value="0" /> </type>`

Many of the attributes control the naming of items. By default, a document may be named GD12345-1A, where various parts of the ID are:

ID element	Description
`GD`	*type (subtype)*
`12345`	*Serial number ID*
`-`	*Delimiter3*
`1`	*Version*
`A`	*Revision*

Complete list of terms that may be used to build item ID is:

`<bizunit>`, `<delimiter1>`, `<subtype>`, `<delimiter2>`, `<number>`, `<delimiter3>`, `<version>`, `<delimiter4>`, `<revision>`

The various terms may be controlled by feature attributes.

6.1. Samples

6.1.0.1. The following sample enables the `Change-undo` button:

```
<type name="doc">
  <feature name="config-manager" change-undo="1" />
</type>
```

The `UNDO-CHANGE` can now be accessed and initiated from any item page within the *Advanced* options section *(requires a minimum set userlevel of AdvancedUser)*:

> **Note:** Progressing the item after a *Change (e.g. sending the Document to Freeze, Review, approval etc.)* will make `UNDO-CHANGE` unavailable, as it can only be used immediately after a `CHANGE` on the item.

The following sample sets *MM* subtype as a review template:

```
<type name="doc">
  <feature name="config-manager" review-template="MM" />
</type>
```

The following sample enables minor revisions

```
<type name="doc">
  <feature name="config-manager" minorRevisions="1" />
</type>
```

The following sample illustrates using alpha version, numeric revision, preceding with *"P"* for preliminary and no delimiter:

```
<type name="doc">
  <feature name="config-manager" alphaversion="1" alpharevision="0" delimiter3="" preliminary1="P" />
</type>
```

The following sample enables parent items to be in review ahead of the children:

```
<type name="doc">
  <feature name="config-manager" min-status-child-enforcement="3" />
</type>
```

The following sample sets a requirement for child items to be frozen before parents can be frozen:

```xml
<type name="doc">
  <feature name="config-manager" min-status-child-enforcement="1" />
</type>
```

7. <feature name="reference-manager"> element

The ***feature*** element with the name ***reference-manager*** allows for complete control over which types of references can be added to items of specific types under certain conditions *(as managed by associated vector roles)*. The *Reference-manager* feature element can be configured on all available basetype and/or subtypes in Highstage.

The feature element is related to the type in which it is declared and specifies:

- Which types of references can be allowed or disallowed to be added as references.
- What conditions must be met for specific users to add references of the appropriate type *(as defined by vector roles)*.

7.1. Available elements and attributes

Attributes

Name	Description	Values	Mandatory
`name`	Mandatory name of element	reference-manager	✓

Elements:

Name	Available attributes	Description
`<profile>`	`name` , `roles` , `types`	Each unique profile specifics which types of items can be managed as references by associated vector roles *(users under specific conditions)*.

7.1.1. <profile> element

Each unique <profile> element are used to define which ***types*** of references can be managed *(added/updated/removed)* by specific ***roles*** *(select users under certain conditions as defined by listed vector roles)*.

Any number of unique <profile> elements can be added.

Attributes

Name	Description	Mandatory
`name`	Unique name of the `<profile>` element.	✓
`roles`	List of vector roles allowed to manage references.	✓
`types`	List of permitted item types to be managed as references *(Note that listed types are case sensitive)*.	✓

Note: Please notice that the listed `types` are case sensitive. For this reason you must ensure that the list of permitted item types are identical *(with regards to upper and lowercase characters)* to what is defined as the name in the type definition.

7.2. Default configurations

Documents

```
<feature name="reference-manager">
    <profile name="default" roles="ActiveResource1;AdminWrite" types="*"/>
</feature>
```

Parts

```
<feature name="reference-manager">
    <profile name="default" roles="ActiveResource1;AdminWrite" types="*"/>
</feature>
```

Devices

```
<feature name="reference-manager">
    <profile name="default" roles="ActiveResource1;AdminWrite" types="*"/>
</feature>
```

Actions

```
<feature name="reference-manager">
    <profile name="default" roles="Resources;AdminWrite" types="*"/>
</feature>
```

Entities *(Projects, Customers, Departments, Manufacturers, Products etc.)*

```
<feature name="reference-manager">
    <profile name="default" roles="Manager;TeamMember;AdminWrite" types="*"/>
</feature>
```

> **Note:** declaring `*` within the types attribute means that you consider all available base- and subtypes *(default and custom types)*.

Please note that using `*` in the types attribute prevents you from creating any additional profiles with potential exceptions to the existing profile as it will be overruled. If you want to create additional profiles, then you may not use `*` in the types attribute - but instead type in the explicit types to be considered.

7.3. Samples

Allowing all users to add references of any type

The following sample permits all types of references to be added to Documents by all users at all times *(regardless of document status)*:

```
<type name="doc">
    <feature name="reference-manager">
        <profile name="default" roles="Everyone" types="*" />
    </feature>
</type>
```

Allowing references of basetypes but excluding specific subtypes

The following sample illustrates how profiles can be configured to allow for various basetypes to be added as references, but exclude specific subtypes:

```
<type name="doc">
    <feature name="reference-manager">
        <profile name="default" roles="Everyone" types="doc;part;device;action" />
        <profile name="excluded" roles="None" types="GD;SOP;GDO" />
    </feature>
</type>
```

In the above sample we allow all users to add/remove/modify references of type Documents, Parts, Devices and Actions.
We then create a new profile *(excluded)* as an exception where the disallow anyone to add/remove/modify references of type GD, SOP and GDO.

> **Note:** If you make multiple profiles, remember that profiles that deal with subtypes will overrule what is specified on profiles that deal with base types. This allows you to make additions and exceptions to the basic settings.

8. Run script on Item change

> Not currently supported

9. Reset column values on item change *(change to Working state)*

In some cases, certain field values should be reset to database default when an item is changed, i.e. set to Working state. For example, the change note should be cleared. It may, however, be discussed if the value should be cleared at any change or just when changed from the Approved state.

The column attribute `change-reset-step` can be used for this purpose.

The value of the attribute is set to the minimum step from which change must reset field value, so if the field must be reset when changing an approved document then the `change-reset-step` value must be set to 5.

The following sample clears the change note when an item is changed i.e. changes state from Approved to Working:

```
<type name="doc">
  <column name="note" change-reset-step="5" />
</type>
```

The First Article Inspection (FAI) sample resets FAI_note at any change `change-reset-step="1"`.

10. First Article Inspection *(FAI)*

The following sample implements a solution for First Article Inspection (FAI):

```
    <type name="part">
      <column name="FAI_check" clone="0" description="" sqltype="bit" sqlnull="0" sqldefault="(1)" />
      <column name="FAI_result" clone="0" write="everyone" change-reset-step="1" values=";OK;FAILED"
  description="" sqltype="varchar" sqlsize="50" sqlnull="0" sqldefault="('')" />
      <column name="FAI_note" virtual="1" column="FAI_note_data" write="everyone" text-add="prepend"
  reload="1" />
      <column name="FAI_note_data" clone="0" change-reset-step="1" write="adminwrite" description=""
  sqltype="ntext" sqlnull="0" sqldefault="('')" />
    </type>
```

Attribute descriptions:

Attribute	Description
`FAI_check`	`FAI_check` is a checkbox always set to 1, but the author may set it to 0, thereby signaling that `FAI_check` is not required.
`FAI_result`	`FAI_result` is a multivalued field that can be set to the result of the inspection.
`FAI_note`	`FAI_note` is a note. Data cannot be deleted, only added to the field, date and users ID is also added at each note addition.
`FAI_note_data`	`FAI_note_data` is the physical column. This column may be written by AdminWrite to correct errors.

11. Note field for adding user comments

In some situations, it is desirable to have the possibility for users to add notes related to an item. The notes could be added any time, by any user, even if the item has been approved.

11.1. Using database column

Columns of SQL type **text** and **ntext** supports the **text-add** attribute with the following values:

Append	Appends text to the end of a database field.
Prepend	Prepends text to start of database field.

When the text field with the text-add attribute is edited, the field text in edit dialog is initially empty. When the user enters text and clicks ok the text is either appended or prepended to the existing field text in the database, the text is also prepended with DateTime and user id. Previously entered text cannot be modified.

For everyone to be able to **write** at any time, the write attribute has to be set to **everyone**.

By setting **cssclass** to **ts-notesizelimit** the field is limited in size on the web page. If field content is very large, then the field is still fairly small and has a vertical scrollbar.

Sample defining new column named NotePad, with **text-add** attribute set to **prepend**:

```
<column name="NotePad"
        description="Note field writeable by everyone at any time. Text entered is prepended to existing
text."
        text-add="prepend"
        write="everyone"
        cssclass="ts-notesizelimit"
        sqltype="ntext"
        sqlnull="0"
        sqldefault="('')"
/>
```

> Remember to apply changes by navigating to SYSTEM > SCHEMA > APPLY

11.2. Using Eventlog

The following implementation defines a virtual column (**comments_**) and adds the column to a subform in **doc** type. Since **part** and **device** types inherit from **doc** type, the field will also be available in these types. New comments are added to the event log, and a section shows all columns related to the item.

1. Copy the following text to the file `comments.schema.xml` in tweak folder:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<schema schemasyntaxversion="0.8">
```

```
  <type name="doc">
    <execute name="comments" src="~/tweak/comments.exe.aspx" />
    <column name="comments_" title="Comments" virtual="1" write="everyone" description=""
sqltype="varchar" sqlsize="2038" sqlnull="0" sqldefault="('')" />
    <form name="default">
      <subform name="comments" x-index="1">
        <field name="Comments_" title="Comments"/>
        <section name="comments" title="Comments" showcount="1" url="~/ts/search.aspx?
t=ts_eventlog&amp;objtype=@[t]&amp;obj==@[o]&amp;field=comments_&amp;_columns=eventtime;userid;Comments"/>
      </subform>
    </form>
  </type>

  <type name="ts_eventlog">
    <column name="obj_filter" virtual="1" column="obj" />
    <column name="Comments" virtual="1" column="NewValue" />
  </type>
</schema>
```

2. Copy the following text to the file `comments.exe.aspx` in tweak folder:

```
<%@ Page Language="C#" validateRequest="false" EnableSessionState="false" enableViewState="false" %>
<%
    TS.Type type = TS.Type.Get(Request.QueryString["t"]);
    TS.Column.Get(type, "comments_").WriteFieldLevel50 = new
TS.Column.WriteFieldLevel50Delegate(CommentsRender);
    TS.Column.Get(type, "comments_").DbWrite  = new TS.Column.DbWriteDelegate(CommentsDbWr);
%>
<script runat="server">
    static void CommentsRender(StringBuilder response, TS.Column column, TS.Xhtml.Field.FieldWriter
writer)
    {
        writer.WriteFieldLevel50StartStandard(column);
        writer.WriteFieldLevel50EndStandard();
    }
    static void CommentsDbWr(TS.Column column, string obj, string oldValue, string newValue, string msg)
    {
        TS.Event.Write(TS.Event.Severity.OK, column.ownerType.baseType.name, obj, msg, "Field",
column.name, "NewValue", newValue);
    }
</script>
```

3. Include `comments.schema.xml` in `custom.schema.xml` by including the following snippet in the custom.schema.xml file:

```
<include src="/tweak/comments.schema.xml" />
```

> Note that the line should be placed ***after*** the line which includes the `highstage.schema.xml`

12. Virtual columns

Columns do not need to be physical database columns. Columns may be virtual. By setting the virtual attribute Highstage does not create a physical column in the database, and depending on other column attributes Highstage may not fetch data from the database either.

12.1. Virtual column as alias to physical column

The following is a sample of creating a virtual column that is “identical” to another physical column. However, it has another name.

The new virtual column name is Author, and it reuses the physical column *EditBy*:

```
    <column name="Author" virtual="1" column="EditBy" />
```

Author will inherit all attributes of *EditBy*. Author may overrule inherited attributes without affecting *EditBy* column. Observe that code behind hooked up on Author column will not be inherited by *EditBy* column.

12.2. Virtual column with SQL select

A column may use SQL select to get data to be displayed in Highstage.

12.2.1. Display parent description

The following sample displays first parent action description:

```
<column name="ParentActionDescription" title="Parent Description" virtual="1" write="0">
    <sqlselect name="default">
      <![CDATA[
        SELECT Description FROM Item WHERE Item=(SELECT TOP 1 Parent FROM ObjRef WHERE ParentType='action'
and ChildType=[.].[ObjType] AND Child=[.].[Item])
        ]]>
    </sqlselect>

  </column>
```

12.3. Dispaly age using sql datediff

This sample displays age of Item as the diff between CreateDate and actual date:

```
    <column virtual="1" name="Age" sqlselect="datediff(day, [.].createdate, getdate())" sqltype="int"
description="Number of days since creation" write="0" />
```

13. Mails

13.1. Send mail from custom code

The following snippet is an example for sending a mail. The parameters are strings:

```
try
{
    TS.Sys.Smtp.Send(subject, body, sourceAddress, destinationAddress);
}
catch (Exception ex)
{
    throw new Exception("SMTP server send error", ex);
}
```

14. Eventlog

14.1. Insert event into Eventlog by custom code

Sample code for logging an event related to an object. Remember to replace `type-name` with `<object-id>` valid values:

```
TS.Event.Write(TS.Event.Severity.OK, "<type-name>", "<object-id>", "This string goes into Eventlog message column");
```

If the event is not related to a specific type, then the type and object can be set to String. Empty or null. This is an example of logging mail send event to Eventlog:

```
TS.Event.Write(TS.Event.Severity.OK, null, null, "This string goes into Eventlog message column ");
```

15. References

15.1. Using *(ref-type)* element to specify grid-columns for specific child type

In some situations, it may be desirable for parent type to specify grid and/or columns for child references of a given type.

By placing this in parent type schema, child parts' will be displayed using the BOM grid *(if the BOM grid has been defined in part type)*:

```
<ref-type name="part" grid="BOM" />
```

By placing this in parent type schema, the child doc's grid will only display item and description columns:

```
<ref-type name="doc" columns="item;description" />
```

The grid and columns attributes may be combined.

The ref-type element can be specified at basetype or subtype level.

15.2. Ability to expand references *(+/-)* in search grid for special types

By default only types based on ts_item has the ability to expand references directly in search grid.

To allow this ability for types based on ts_entity add the following snippet to custom.schema.xml:

```
<type name="ts_entity" types="ts_ref" />
```

The types attribute specifies all referenced types to be listed in search grid, however ts_ref is the placed leftmost in grid with +/- button. Other types would be placed rightmost in grid.

15.3. Reference filter and refilter

> The use of refilter should be avoided on basetypes.

Highstage is supplied out-of-the-box with no use of refilter. Raw filter is used for all references. The reason is that no references should be hidden. Even if references are obsolete, not latest, closed or any other reason they should always be visible in reference section. Raw filter shows the truth which may be visible any other place. References may be exported to suppliers or customers, and if they are visible there then they should be visible here as well.

The reason to use refilter may be to only show open actions. This may be achieved by using the object element search-parameters attribute.

16. Roles

16.1. Trustees on roles

By default only DefaultTrustees defined by parameter are able to see all users. This may be important in cases when customers and/or suppliers have access to the system and these users may not be able to see each other’s existence.

In some situations, when customers or suppliers have no access to the system it may however be desirable to have all users have the ability to see each other, in this case the following xml snippet may be added to custom.schema.xml:

```
<type name="ts_role">
  <column name="Trustees" sqldefault="(';Everyone;')" />
</type>
```

Set Trustees on existing roles to Everyone, either by multi-edit or by executing the following SQL update:

```
UPDATE ts_role set trustees=';Everyone;'
```

17. Place specific subtype in specific workspace

By placing the following schema snippet in subtype schema the user will be guided to place item in workspace 000:

```
<?xml version="1.0" encoding="utf-8"?>
<schema schemasyntaxversion="0.8">
  <type name="SLA">
    <form name="create">
      <field name="workspace" value="000" />
      <xhtml _before="workspace" name="workspaceinfo">
        <div class="warning notification large">Please do not change workspace unless required</div>
      </xhtml>
    </form>
  </type>
</schema>
```

The following snippet will enforce subtype to be placed in workspace 000, only SuperUsers will be able to change workspace after item is created:

```
<?xml version="1.0" encoding="utf-8"?>
<schema schemasyntaxversion="0.8">
  <type name="SLA">
    <column name="workspace" write="superuser" />
    <form name="create">
      <field name="workspace" value="000" write="0" />
    </form>
  </type>
</schema>
```

18. Form title customization

The form title may be changed by code behind. The following sample shows how to include workspace and in case of action items also the subtype on first title line, and in second line the subtype title and item title.

18.1. Custom.schema.xml

Include this xml snippet in custom.schema.xml:

```xml
<type name="ts_item">
  <execute name="item" src="~/tweak/item.exe.aspx" />
</type>
```

18.2. Code behind

Include the following code in file `item.exe.aspx` located in TSWebApp/tweak folder:

```
<%@ Page Language="C#" validateRequest="false" EnableSessionState="false" enableViewState="false" %>
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Xml" %>
<%
    TS.Type type = TS.Type.Get(Request.QueryString["t"]);
    type.RenderFormTitle += new TS.Xhtml.Form.FormBuilder.RenderFormTitleDelegate(RenderFormTitle);
%>
<script runat="server">
    static void RenderFormTitle(ref string xhtmlPageTitle, StringBuilder xhtmlFormTitle,
TS.Xhtml.Form.FormBuilder formBuilder)
    {
        xhtmlFormTitle.Length = 0;
        xhtmlFormTitle.Append("<div class=\"ts-title ts-form-title title\">");

 xhtmlFormTitle.Append(System.Web.HttpUtility.HtmlEncode(formBuilder.dataRow["workspace"].ToString()));
        if (formBuilder.type.baseType.name == "action")
        {
            xhtmlFormTitle.Append("/");
            xhtmlFormTitle.Append(System.Web.HttpUtility.HtmlEncode(formBuilder.type.name.ToUpper()));
        }
        xhtmlFormTitle.Append(": ");
        xhtmlFormTitle.Append(System.Web.HttpUtility.HtmlEncode(formBuilder._obj));
        xhtmlFormTitle.Append(" ");
        string subtypeTitle = TS.Column.StandardDbReadCache(TS.Column.Get(TS.Type.Get("ts_subtype"),
"name"), formBuilder.type.name).ToString();
        string title = (string)formBuilder.dataRow[formBuilder.type.titleColumn];
        xhtmlFormTitle.Append("<br/>");
        xhtmlFormTitle.Append(System.Web.HttpUtility.HtmlEncode(subtypeTitle));
        xhtmlFormTitle.Append(": ");
        xhtmlFormTitle.Append(System.Web.HttpUtility.HtmlEncode(title));
        xhtmlFormTitle.Append("</div>");
    }
</script>
```

19. Advanced detailed menu and main page permissions

To restrict access to certain elements like section and links on menu and main page the trustees attribute may refer to a specific field in database containing list of trustees.

One build-in sample using this approach is the Create type (subtype) link for items like actions, documents, parts and devices. The trustees attribute refers to TrustCreate on the ts_subtype basetype.

```
<link name="Create type" url="ts/new.aspx?t=ts_subtype&amp;basetype=@[t]"
trustees="ts_basetype:trustcreate:ts_subtype" />
```

Similar if we imagine Highstage used as a contract archive where all contracts are stored on a confidential `LEGAL` workspace, then the menu could contain a menu that is only visible to LEGAL workspace trustees:

```
      <section name="Contracts" trustees="ts_entity:trustees:LEGAL">
        <link name="search" url="ts/search.aspx?t=CONTRACT" />
        <link name="New" url="ts/new.aspx?t=CONTRACT" />
      </section>
```

The main page could contain a section with contracts to process:

```
      <section name="CONTRACTS" title="CONTRACTS" expanded="1" url="ts/search.aspx?
t=CONTRACT&amp;_grid=todo" trustees="ts_entity:trustees:LEGAL" />
```

The create form for contracts would ensure that all new contracts are forced to LEGAL workspace by setting LEGAL as workspace value, and hiding the field from user or disabling changing field value by setting write attribute to 0:

```
    <form name="create">
      <field name="workspace" value="LEGAL" hidden="1" write="0" />
    </form>
```

20. Create automation

20.1. Specify suppliers in create form and add references on create

Define virtual column suppliers in `custom.schema.xml`, and add it to create form. Also add an execute to the file `doc.exe.aspx` which is defined below:

```
<type name="doc">
    <column name="suppliers" virtual="1" type="supplier" objlist="1" />

    <form name="create">
      <field name="suppliers" />
    </form>

    <execute name="doc.exe.aspx " src="~/tweak/doc.exe.aspx" />
  </type>
```

Create the file doc.exe.aspx with the method `OnAfterCreate`:

```
<%@ Page Language="C#" validateRequest="false" EnableSessionState="false" enableViewState="false" %>
<%@ Import Namespace="System.IO" %>
<%@ Import Namespace="System.Xml" %>
<%
    TS.Type type = TS.Type.Get(Request.QueryString["t"]);
    type.OnAfterCreate += new TS.Type.OnAfterCreateDelegate(OnAfterCreate);
%>
<script runat="server">
    static void OnAfterCreate(TS.Object.Creator creator)
    {
        System.Xml.XmlNode node = creator.objectNode.SelectSingleNode("field[@name='suppliers']");
        string suppliers = node.InnerText;
        string newObj = (string)creator.dr[creator.type.keyColumn.name];
        foreach(string s in suppliers.Split(';', ','))
        {
            string supplier = s.Trim();
            if (String.IsNullOrEmpty(supplier)) continue;
            try
            {
                TS.Ref.Add(creator.type.baseType.name, newObj, "supplier", supplier);
            }
            catch { }
        }
    }
</script>
```

21. Refinery

21.1. Disable pdf file creation in Excel refine

Pdf rendition may be disabled by adding the following snippet to custom schema:

```
<type name="_file">
  <refinery name="file">
    <app name="ms_excel" rendition="" />
  </refinery>
</type>
```

21.2. Disable refine on Excel files

To disable Excel file refine, so that excel files are not touched by Highstage refinery, add the following code snippet to custom schema:

```
<type name="_file">
  <refinery name="file">
    <app name="ms_excel" _delete="1" />
  </refinery>
</type>
```

21.3. Add properties directly in schema without code

New properties may be added like in the following snippet to custom.schema.xml:

```
<type name="_file">
  <refinery name="file">
    <property name="test1">This is test1 property</property>
    <property name="Area">@[data:area]</property>
  </refinery>
</type>
```

21.4. Folder refine

Folder refine is executed after all file refines has been executed.

The following are two folder refine samples FolderRefine1 and FolderRefine2. FolderRefine2 has the ability to be executed by user as well.

```
<type name="doc">
  <refinery name="folder">
    <execute name="folderrefine1" src="~/tweak/folderrefine/folderrefine1.exe.aspx" />
    <execute name="folderrefine2" src="~/tweak/folderrefine/folderrefine2.exe.aspx" />
  </refinery>
</type>
```

FolderRefine1.exe.aspx:

```
<%@ Page Language="C#" validateRequest="false" EnableSessionState="false" enableViewState="false" %>
<%
    //Copy this file to tweak folder and add the following schema snippet to custom.schema.xml:
    //  <type name="doc">
    //    <refinery name="folder">
    //      <execute name="folderrefine1" src="~/tweak/folderrefine1.exe.aspx" />
    //    </refinery>
    //  </type>



    //This sample writes folderrefinetest1.txt file to item folder
    //Exception handling is not done since it is handled at outer level
    //It has been prohibited to call thes .aspx directly by users, it must be called by web server

    TS.Core core = TS.Core.Current;
    if (core == null)
    {
        Response.Write("This .aspx can not be called directly by user.");
        return;
    }
    string typeName = Request.QueryString["t"];
    TS.Type type = TS.Type.Get(typeName);
    string obj = Request.QueryString["o"];

    System.Data.DataRow dr = TS.Sql.ExecuteDataRow("SELECT FolderRelativePath FROM " +
type.sqlTableFullName + " WHERE " + type.keyColumn.sqlColumnName + "='" + obj + "'");
    string folderPath = System.IO.Path.Combine(TS.dParameter.StoragePathServer,
(string)dr["FolderRelativePath"]);

    string filePath = System.IO.Path.Combine(folderPath, "folderrefinetest1.txt");
    using (System.IO.StreamWriter sw = System.IO.File.CreateText(filePath))
    {
        sw.Write("This file is produced by folder refine sample folderrefine1.exe.aspx");
    }
%>
```

FolderRefine2.exe.aspx:

```
<%@ Page Language="C#" validateRequest="false" EnableSessionState="false" enableViewState="false" %>
<%
```

```
    //Copy this file to tweak folder and add the following schema snippet to custom.schema.xml:
    //  <type name="doc">
    //    <refinery name="folder">
    //      <execute name="folderrefine2" src="~/tweak/folderrefine2.exe.aspx" />
    //    </refinery>
    //  </type>



    //This sample writes folderrefinetest2.txt file to item folder
    //Exception handling is not done since it is handled at outer level
    //May be run by user, so uses PageBuilder to make everything look and feel right

    TS.Xhtml.PageBuilder page = new TS.Xhtml.PageBuilder();
    page.Begin("folderrefine2");

    string typeName = Request.QueryString["t"];
    TS.Type type = TS.Type.Get(typeName);
    string obj = Request.QueryString["o"];

    System.Data.DataRow dr = TS.Sql.ExecuteDataRow("SELECT Item, Description, status, FolderRelativePath
FROM " + type.sqlTableFullName + " WHERE " + type.keyColumn.sqlColumnName + "='" + obj + "'");
    string folderPath = System.IO.Path.Combine(TS.Parameter.StoragePathServer,
(string)dr["FolderRelativePath"]);
    int status = (int)dr["status"];

    int exitCode;
    string stdout;

    //The sample below could be improved by checking exitCode

    //Sample of calling cmd.exe and execute dir command to list directory
    TS.Sys.Command.Execute(System.Environment.SystemDirectory + @"\cmd.exe", " /c dir /s "+folderPath,
out exitCode, out stdout);

    string filePath = System.IO.Path.Combine(folderPath, "folderrefinetest2.txt");
    using (System.IO.StreamWriter sw = System.IO.File.CreateText(filePath))
    {
        sw.Write((string)dr["item"] + " " + (string)dr["description"]);
        sw.Write("\r\n");
        sw.Write("Item has status: " + status);
        sw.Write("\r\n");
        sw.Write("This file is produced by folder refine sample folderrefine2.exe.aspx");
        sw.Write("\r\n");
        sw.Write(stdout);
    }

    page.Append("The following has been written to folderrefinetest2.txt:<hr/>");
    page.Append(Server.HtmlEncode(stdout).Replace("\n","<br/>"));

    page.End();
%>
```

22. Item change permissions

By default the following vector roles has permissions to change an item:

```
<type name="doc ">
  <feature name="config-manager" change-write="resource1;manager;teammember;superuser"/>
</type>
```

This may be modified to include for example DefaultTrustee:

```
<type name="doc ">
  <feature name="config-manager" change-write="resource1;manager;teammember;superuser;defaulttrustee"/>
</type>
```

23. Moving subforms using x-index and y-index

Subforms may be moved relatively to each other by changing x-index and/or y-index attributes. Since it is relatively to other subforms, it is good to know the x-index and y-index values of the other subforms. This can be investigated in two ways:

1. In the merged schema in menu System/Types SchemaXml column rightmost:

2. By System/Trace/Execution:

x-index and y-index values can be positive and negative, and they can be floating points.

Even though x-index and y-index values can be set to very large values the forms won’t be as large. The x-index and y-index are just used to position subforms relative to each other, so subforms with very big values will just be positioned far right or at the bottom.

Default x-index and y-index values are 0, and the default subform position behavior is to place the subforms in the order they appear in schema.

23.1. Sample

Move Document options subform down below the document properties:

```
<type name="doc">
    <form name="default">
        <subform name="right" y-index="2" />
    </form>
</type>
```

24. Make a grid column narrow

The column attribute `header-cssclass` together with a css snippet can do it.

Sample making EventTime grid column narrow:

xml snippet in custom.schema.xml:

```xml
<type name="ts_item">
  <column name="eventtime" header-cssclass="narrow-column" />
</type>
```

css snippet in custom.css:

```css
.narrow-column{max-width:60px;}
```

25. Show image/picture from item folder on web page

If image is only required on item form, then this schema snippet added to form element is a very simple solution showing a well-defined image file name on web page:

```
<xhtml name="toolpicture">
    <img
        name="toolpic"
        src="@[parameter:StorageUrl]/@[data:FolderRelativePath]/tool.png"
        alt="Missing picture"
        width="120"
        height="120"
    />
</xhtml>
```

Alternatively, the functionality could be added as a column with a cform so that image can appear in searches:

```
  <type name="ts_item">
    <column name="Toolpic" virtual="1">
      <cform name="default">
        <xhtml name="default">
          <img  name="toolpic"
src="@[parameter:StorageUrl]/@[data:FolderRelativePath]/tool.png"
alt="Missing picture"
width="120"
height="120"
         />
        </xhtml>
      </cform>
    </column>
  </type>
```

If the following schema is added to schema, then the image will be zoomed when cursor hoovers image:

```
  <js name="jQueryImgZoom" src="ts/jquery/plugin/imgZoom.js">
    <![CDATA[
$(document).on('T_onload', function(event, target){
  $(function () { $(target).find("img[name='toolpic']").imgZoom({  zoomSize: '400px' }); });
});
  ]]>
  </js>
```