Document name	Revision
DOCU12198	1

1. Turbostage Schema reference

1.1. Background

Turbostage Schema is inspired by best-practice object-relational mapper systems by Apache and Microsoft.

It is a system which may spread across several databases and servers. Instead of using more schema layers to define the systems TurboStage defines the complete system in one layer.

Complex business logic is contained in code hooked up to the schema using the <execute> element, the code is however referenced in the schema making the schema covering all aspects – in one layer.

1.2. Definitions

Name	Definition
`List`	A list in TurboStage terminology is a list of names separated by semicolon.

1.3. Schema merging

1.3.1. Basis

The schema files define the data architecture and functionality. The only file that in fact is loaded by
TurboStage is `custom.schema.xml` located in the tweak folder.

This root schema file may include other schema files.

All schema files must start with the <schema> tag. The following sample shows custom <schema> element. The `schemasyntaxversion` attribute may not be altered manually, it is reserved for public use by TurboStage to keep track of schema version and schema upgrade.

```
<schema schemasyntaxversion="0.7">
```

You may design your own system from scratch, however some required types has to be defined in order for TurboStage to startup.

These minimum requirements are defined in `/ts/schema/minimum.schema.xml` which you may include and build your system from.

```
<include src="/ts/schema/minimum.schema.xml" />
```

1.3.2. Include files

The normal procedure for using TurboStage as a business system covering portal and PDM functionality
is to just include the Highstage schema file which defines types like Action, Document and Part.

For schema backward compatibility with old Highstage generation use:

```
  <include src="/schema/highstage/highstage.schema.xml" />
```

Schema files are merged together in the order they are included. If a schema file file1 includes file2
then file2 is loaded first and then file1 is merged to produce the resulting schema xml.

1.3.3. Element merging

Merging of two schema files will be done by adding all elements to the resulting schema.
However if two elements exists with the same element name then the elements will be merged.

Sample

```
<!--Schema file 1:-->
    <element name=”x” />
<!--Schema file 2:-->
    <element name=”y” attribute=”a” />
<!--Schema file 3:-->
    <element name=”x” attribute=”b” />
<!--Resulting schema:-->
    <element name=”x” attribute=”b” />
    <element name=”y” attribute=”a” />
```

> Note that it is important that **all elements must be given names** . If elements with no names exist, then an exception will occur at system reset.

1.3.4. Merge attributes

For all elements, the following XML merge attributes are available:

Name	Values	Description
`_before`		Inserts node before the node specified by name, instead of appending to the end.
`_delete`	0/1	Removes the xml node completely. The node is deleted after the schema has been merged, so the `_delete` attribute may be altered at any place in schema and the resulting value will determine if the node is deleted or not.
`_inherit`		Inherits the specified sibling node(s) specified in name list before merging the content. The inherited node(s) must precede the node.
`_overwrite`	0/1	Replaces the xml node and attributes with completely new content

1.3.5. Semicolon separated attribute value list

Standard behavior is that when two elements are merged the attributes from the second element in overwrite the attribute values of the first element. However if the second elements attribute starts with `"+="` then the attribute value will be concatenated to the first elements attribute value.

Sample:

```
<!--Merge element 1:-->
     <element name=”x” attribute="a"/>
<!--Merge element 2:-->
     <element name=”x” attribute=”b” />
<!--Merge element 3:-->
     <element name=”x” attribute=”+=c;d” />
<!--Resulting element:-->
    <element name=”x” attribute=”b;c;d” />
```

1.3.6. Inheritance

Inheritance is resolved by first merging leaf's and then merging each level until the root is reached.
If one type inherits from several other types then merging is done from left to right.

1.4. Merge variables

Merge variables `@@[<variable-name>]` may be used in schema xml files that are included by other schema xml files. The <include> element documented later in this document will replace merge variables with attributes in the <include> element. Merge variables that do not have a corresponding <include> element attribute with identical name will result in schema load exception.

1.5. Runtime variables

Runtime variables `@[<variable-name>]` may be used various places in schema xml, refer to element documentation later in this document.

Name	Description
`@[t]`	Type (or subtype)
`@[userid]`	User ID
`@[v]`	Value
`@[/<LocalPath>]`	Variable starting with `'/'` will be resolved as a web page relative to web site root. Page output is returned as result. The web page may make use of `form`, `grid` and `dr` objects passed as `Context.Items` .

1.6. <schema> element

Starting with the <schema> element the following attributes and elements are available:

Attributes:

Name	Description
`schemasyntaxversion`	For public TurboStage use
`productname`	For public TurboStage use
`productversion`	For public TurboStage use

Elements:

Name	Description
`execute`	This element defines an page to execute after the schema has been loaded. This allows for performing specific custom code at startup or setting delegates to custom code.
`type`	Defines an object type
`include`	Includes an external schema file
`job`	Sets a job to be executed. This may execute a web page at regular intervals or at a specific time

1.6.1. <execute> element

This element specifies a page to execute after the schema has been loaded.
This allows for performing specific custom code at startup or setting delegates to custom code.

Attributes:

Name	Description
`name`	Name of element instance
`src`	Relative path to local web page

Sample loading folder feature:

```
<execute name="builtin" src="~/ts/feature/folder.aspx" />
```

1.6.2. <css> element

Includes html cascading style sheet (CSS).

Attributes:

Attributes	Description
`name`	Name of element instance
`src`	Location of file

Sample:

```
<css name"style" src"/tweak/style.css" />
```

1.6.3. <include> element

The include element loads additional schema xml files.

Attributes:

Name	Description
`name`	Name of element instance
`src`	Path to source file *(See sample below)*
`<parameters>`	Parameters may be specified *(See sample below)*
`_remove`	List of xpath expressions selecting notes to be removed. Use semicolon `;` char as spath expression delimitter.

Samples on src attribute:

Path relative to the folder containing the current schema file:

```
<include src="sample.schema.xml" />
```

Absolute path:

```
<include src="c:\...\sample.schema.xml" />
<include src="\\server\share\...\sample.schema.xml" />
```

Path relative to web application root folder:

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

Samples on <parameters>:

All attributes of the include element may be used as parameters in the include file:

```
<include src="/schema/axapta.schema.xml" typePrefix="AXAPTA:"
tablePrefix="AXAPTA.dbo." sqlmanage="0" dataareaid="DAT"/>
```

> The value of attribute `typePrefix` will replace all occurrences of `@@[typePrefix]`.

Samples on <_remove>:

Include schema imported from HS, but we are mainly interested en column definitions and want to remove all menu, main and form nodes:

```
  <include src="HS_Imported.schema.xml" _remove="//main;//menu;//form" />
```

1.6.4. <job> element

The job runs specific web pages at a specific interval or time.

Attributes:

Name	Description
`name`	Name of element instance
`url`	Url to execute
`interval`	HH:MM interval specified in 24 hour format. Minimum 00:00 and maximum 23:59.
`time`	HH:MM specified in 24 hour format. Minimum 00:00 and maximum 23:59. Multiple times may be specified as a semicolon separated list of HH.MM times.

The interval and time attributes may be combined but normally just one of them are used.

Sample:

The following sample runs the local server page replicate.aspx at 1 hour interval:

```
  <job name="LdapReplicate0100I" interval="01:00" url="/ts/ldap/replicate.aspx"
/>
```

1.6.5. <js> element

Includes JavaScrips file and/or inline JavaScript.

Attributes:

Name	Description
`name`	Name of element instance
`src`	Location of file

Sample:

```
<js name="jquerytextareaautosize" src="ts/jquery/plugin/jquery.autosize-min.js">
    <![CDATA[ $(function () { $('textarea').autosize(); }); ]]>
 </js>
```

1.6.6. <type> element

The type element defines a specific object type.
Refer to Schema_type_element.htm for more information on the <type> element.

1.7. Advanced process security

Refer to TN10682-1.pdf

1.8. Integrating foreign systems

Foreign systems are integrated by creating a schema for the system or the parts of the system that are relevant for integration. As an example ERP systems typically contains many tables (up to many thousands) of which only a few tables (~10) are relevant for the daily core business.

If the foreign system tables are not contained on the same SQL server as TurboStage then the SQL server must be linked to the TurboStage SQL server and the foreign system tables must be referenced using a fully
qualified path where <server> is the name of the linked server:

```
<server>.<database name>.<schema name>.<table name>
```

Sample of a fully qualified path:

```
sql1.axapta.dbo.custtable
```

> **IMPORTANT**
> The foreign system types should typically have sqlmanage attribute set to '0' to eliminate risk of redefining foreign SQL tables. The user name used for linked server login should typically have read-only access.