XPath Reference

FOX:Schema:XPATH REFERENCE

- XPath String
- XPath Boolean
- Complex XPath
- Simple XPath
- Contexts

FOX:Reference:XPath String

XPath String

XPath Strings can be either a string literal or a Complex XPath which evaluates to a string. If using the latter form, the XPath must be wrapped in the string() function to indicate to FOX that it should evaluate the expression.

Examples

```
 "This is a literal XPath String."
```

"string(:{root}/STRING_ELEMENT/text())"

FOX:ReferenceXPath Boolean

XPath Boolean

XPath Booleans are a Complex XPath which will be cast to a boolean.

FOX:Reference:Complex XPath

Complex XPath

A **Complex XPath** is an XPath expression which may contain FOX-specific context markup as well as XPath predicates and axes. Like Simple XPaths, Complex XPaths follow the evaluate context rule.

A Complex XPath can return a Node, Node List, Boolean, or String.

Default Contexts

The following default contexts are available to any Complex Xpath expression:

- :{attach}
- :{baseself}
- :{error}
- :{params}
- :{root}
- :{theme}
- :{result}
- :{return}
- :{session}
- :{sys}
- :{temp}
- :{user}

XPath Extensions

exists-context

```
exists-context(:{contextName})
```

This can be used to test if a given context exists.

FOX:Reference:Simple XPath

Simple XPath

A **Simple XPath** is a form of XPath which cannot use FOX-specific DOM contexts, such as :{root}. All other XPath language constructs are available to it.

Simple XPaths are often used in a relative manner by a command. For instance the fm:init command uses a relative Simple Xpath in its `for-schema` attribute.

FOX:Reference:Contexts

Contexts

Contexts are a FOX-specific XPath enhancement available in all Complex XPaths. A context is a pointer to a specific node in a DOM. The exact contexts available to a Compex XPath depend on the command or function processing the XPath.

Note: to test if a context exists you can use exists-context.

DOM

The FOX DOM contexts are always available. They are immutable and always point to the root node of a built-in FOX DOM.

:{error}

Scope	Description
1 Module Call	Contains a list of validation errors, generated when the fm:validate command is run. You should not write data to it.

:{params}

Scope	Description
1 Module Call	Contains any parameters passed to the module from when it was called. You should not write data to it.

:{prefs}

Full implementation unknown as of July 2011. Coming soon to a FOX Open build near you!

:{result}

Scope	Description
1 Module Exit	This is where anything placed into the :{return} DOM can be accessed in the module next in the module stack (once the top module has been exited), usually as part of a callback action.

For example, after a FOX module has called the DEC044X module (Person Search) and selected a person (i.e. returned back to the calling module), the :{result} DOM could look like:

```
<result>
  <P_ID>12345</P_ID>
</result>
```

and the P_ID element within it can be accessed via the XPath expression:

```
:{result}/P_ID.
```

:{return}

Scope	Description
1 Module Call	Anything placed into the :{return} DOM will be available in the :{result} DOM of a calling module upon a module exit.

For example, if a FOX module has called the DEC044X module (Person Search), this module may place a P_ID element within the :{return} DOM in order to pass the P_ID element up to its calling module once it has been exited.

:{root}

Scope	Description
1 Module Call	This is the main area of data storage and is written to the database every time an external action (i.e. an HTML form post) is run, using the Storage Location as defined in the module's current Entry Theme.

:{session}

Scope	Description
1 Browser Session	Used for one Browser session, that is for the life of one Browser window. It is useful for storing copied data to be pasted into different modules. NOTE: current versions of FOX only store the session DOM in memory. If FOX is restarted, all session DOMs are lost. This issue is fixed in an as yet un-numbered future FOX release.

:{env}

Scope	Description
1 Module Stack	Similar to the session DOM but allows FOX to circumvent the restriction of one session per browser window. It is useful for copying data forwards. The :{env} DOM behaves as a stack, with data placed into it available to the module itself and any modules called by it (and modules called by those, etc.).

:{sys}

Scope	Description
Persistent	Contains useful system information pertaining to the state of the FOX engine and callstack, including database SYSDATE and SYSDATETIME, and the name and Entry Theme of the current Module.

:{theme}

Scope	Description
1 Module Call	Similar to the Root DOM, the Theme DOM exists for the duration of a module call. However it is not possible to save the contents of the Theme DOM directly to the database. As such, it is useful for generating presentational or metadata elements which are safe to discard after a module is exited.

:{temp}

Scope	Description
1 Transaction Process Cycle	Used to set values or collate data from multiple sources to be used in one place for a transaction, but its life span is very short and cannot be used for setting out data

:{user}

Scope	Description
1 Portal Session	Stores information about the current user logged in to the UK Oil Portal. You should not write information to it.

Predefined

:{attach}

This always points to the current attach point and is the same as using a '.', with the benefit that it can be used in predicates.

:{action}

This points to the parent node of the node on which the action was declared.

For example:

```
<xs:element name="PARENT_NODE" main:edit=".">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="phantom-dummyAction" type="phantom"
```

```
actions:ro="." actions:run="." fox:prompt="Do Action"
fox:action="action-dummyAction" fox:widget="button"/>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
```

If the phantom-dummyAction element is clicked then :{action} will evaluate to PARENT_NODE.

:{assignee}

Used with fm:assign when you are assigning a value using a setTarget and an XPath, the :{assignee} context refers to the original value of the element. This means you can use reference the original value within your assignment.

For example:

```
<fm:assign initTarget="/*/ITEM/PRICE" expr="round(:{assignee} div 100 * 105 * 100) div 100"/>
```

:{item}

Within fox:change-actions

Used to reference the element that triggered a fox:change-action within the called action. This can, for example, be used to perform more sophisticated (possibly dynamic) validation on an element, or to set flags.

If the below element exists (and is set-out) and is changed by the user then the change-action will be run at the start of the next page request. During this action :{item} will evaluate to FOO.

```
<xs:element name="SOME_DETAIL" aNamespace:ro=".">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="FOO" aNamespace:edit="." fox:change-action="action-SOME_MODULE-the_change_action"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

An example of how the action could behave.

```
<fm:action name="action-SOME_MODULE-the_change_action">
  <fm:assign initTarget=":{temp}/CURRENT_FOO_IN_METRES" expr="concat(:{item}/text(),'m')"/>
</fm:action>
```

:{itemrec}

The :{itemrec} context is for convenience, and is the equivalent of

```
:{item}/..
```

Within fox:change-actions

Used to reference the complex element that triggered a fox:change-action within the called action. This can, for example, be used to perform more sophisticated (possibly dynamic) validation on an element, or to set flags.

If the below element exists (and is set-out) and is changed by the user then the change-action will be run at the start of the next page request. During this action :{itemrec} will evaluate to SOME_DETAIL.

```
<xs:element name="SOME_DETAIL" aNamespace:ro=".">
  <xs:complexType>
```

```
    <xs:sequence>
      <xs:element name="FOO" aNamespace:edit="." fox:change-action="action-SOME_MODULE-the_change_action"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

An example of how the action could behave.

```
<fm:action name="action-SOME_MODULE-the_change_action">
  <fm:run-query db-interface="dbint-SOME_MODULE-validation" query="qry-validate_foo" match=":{itemrec}"/>
  <!-- query sets :{temp}/FOO_VALID to 'Y' or 'N' -->
  <fm:if test=":{temp}/FOO_VALID = 'N'">
    <fm:then>
      <!-- perform actions if FOO is invalid -->
    </fm:then>
  </fm:if>
</fm:action>
```

:{baseself}

:{baseself} is equivalent to . outside predicates. Within a predicate it can be used to refer to the 'base' element of the XPath expression. This is useful for back-referencing within predicates as XPath does not natively provide a method for doing this. For example for

```
fox:validate-xpath="./A/B[. = 1 or . = :{baseself}/C]"
```

the '.' within the predicate refers to element B. The :{baseself} context refers to the XPath's evaluate context (the initial '.' in the outer statement)