2015

Virtual XML 1.0a

VIRTUAL XML DATA STORAGE

Doc Rev 1.01

mailto: ivvvsx@gmail.com

I. Basics

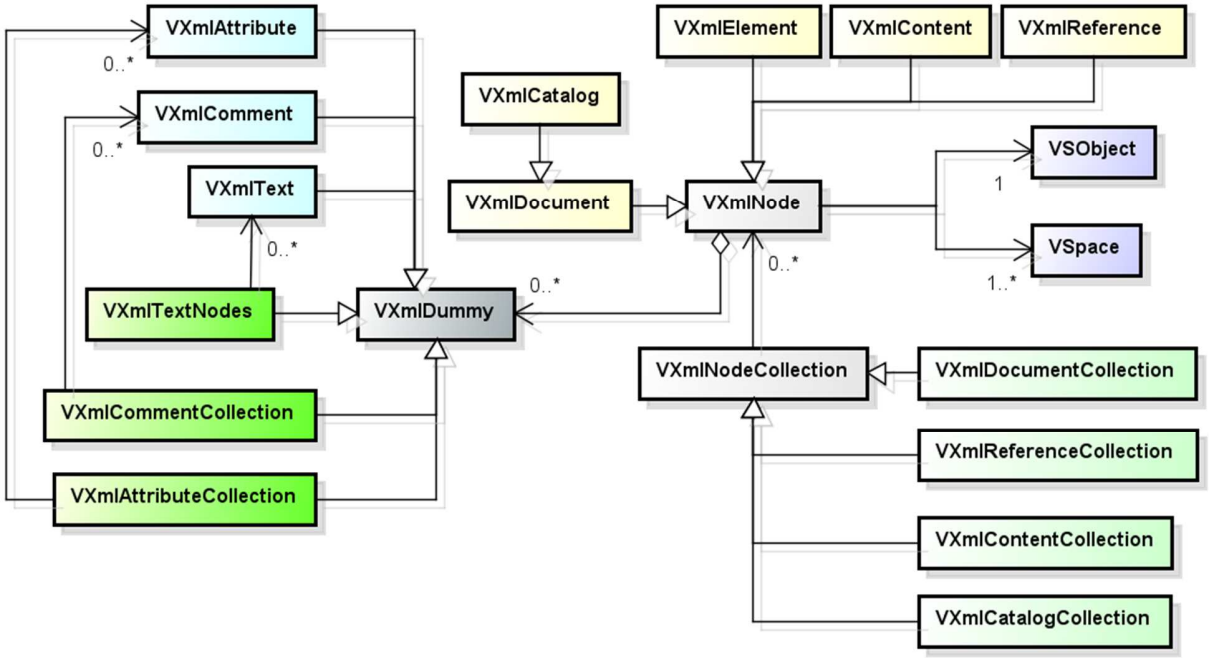
VXML (Virtual XML) is XML-based database built on Virtual Storage platform as VStorage wrapper.
It inherits all existing Virtual Storage capabilities, such as distributed allocation of the database files, multi-space approach for specific data pools, dump/restore functions, static and dynamic storage extension, transactions, physical data consistency, etc.
VXML database is NOT a single XML document. It provides the embedded catalog structure that can be managed by user in the specific manner, XML documents are attached to catalog nodes. XQL queries may retrieve XML nodes from the specific document/element or documents list from the catalog structure using internal documents nodes as well as document names in the queries.
So, there are some specific extensions in VXML:

- New node type ‘Catalog’ that allows build the catalog tree.
- New node type ‘Reference’ that allows create reference to the ‘Catalog’ or ‘Document’ objects. References could be used instead of cloning real objects.
- New node type 'Content' that allows create binary content attached to the document element. Binary content could be stored in the separate Virtual Storage space for easy management. Anyway the data consistency will be guaranteed by the Virtual Storage cross-space transactions.
- Extended XQL language to search objects within catalog tree as well as within particular document; multi-level search conditions could be defined.
- Data portability. Any document or document element (including all subtree) could be exported using CheckOut method to the file in the internal portable format with assigning unique GUID identifier. This file can be checked in to another database and the source data will become read-only. This capability allows safe data exchange between two or more VXML databases.

VXML also provides its own UI administration tool (VXmlExplorer) that allows manage data spaces for VXML storage as well as individual objects within the database (create, edit, delete).

II. Architecture and Design

1. VXML Object model



1.1 Nodes

Virtual XML includes the root class (VXmlNode) that encapsulates most of the node properties and methods and set of the descendant classes representing the specific of the particular node types.

VXmlNode – root node class that contains the core VXML node properties and methods.

VXmlElement – class that represents node type 'element'. Inherited from **VXmlNode**.

VXmlDummy – base class that represents internal node types.

VXmlAttribute – class that represents node type ‘attribute’. Inherited from **VXmlDummy**.
VXmlComment – class that represents node type ‘comment’. Inherited from **VXmDummy**.
VXmlText – class that represents node type ‘text’. Inherited from **VXmlDummy**.
VXmlContent – class that represents node type ‘content’. Inherited from **VXmlNode**.
VXmlReference – class that represents node type ‘element’. Inherited from **VXmlNode**.
VXmlDocument – class that represents node type ‘document’ for XML document. Inherited from **VXmlNode**.
VXmlCatalog – class that represents node type ‘catalog’ for XML document container. Inherited from **VXmlDocument**.

1.2 Collections

1.2.1 Nodes

Virtual XML includes the root class **VXmlNodeCollection** that encapsulates most of the properties and methods for node collections and set of the descendant classes representing the specific of the particular node types.
The number of nodes in the collection can be obtained using ‘Count’ property.
Each node in the collection can be referenced by index (0 … Count-1) or node name (case sensitive).
Collection is returned by XQL query (SelectNodes method) or by node properties:

Node property	Collection class
VXmlNode.ChildNodes	VXmlNodeCollection
VXmlNode.ContentNodes	VXmlContentCollection
VXmlCatalog.ChildNodes	VXmlCatalogCollection
VXmlCatalog.Documents	VXmlDocumentCollection
VXmlCatalog.References	VXmlReferenceCollection

1.2.2 Dummy nodes

Dummy nodes are not separate objects within storage. They are inherited from **VXmlDummy** class.
Collections are also inherited from VXmlDummy:

Node property	Collection class
VXmlNode.Attributes	VXmlAttributeCollection
VXmlNode.CommentNodes	VXmlCommentCollection
VXmlNode.TextNodes	VXmlTextCollection

2. Virtual Storage integration

To open VXML storage you must create root ‘catalog’ node object and specify the root Virtual Storage folder as parameter, e.g.:

```
catalog = new VXmlCatalog("c:\\data\\my_xml_storage");
```

VXML uses three Virtual Storage spaces:

vxmlbase - contains all VXML nodes data and (optionally) binary content;

vxmlcontent - contains binary content of the 'content' nodes; if this space is missing then all information is stored in the **vxmlbase** space.

vxmlindex - contains supplementary indexes.

Space(s) must exist at this point. You can create VXML spaces in VXML administration tool UI (VXmlExplorer), Virtual Storage administration tool (command-line: VSUtil, UI: VStorageExplorer) or directly using Virtual Storage API ('Create' method).

The root catalog object initializes Virtual Storage spaces.
Each node object in VXML is attached to the corresponding 'VStorage.VSObject' and uses VSObject space allocation to store all node information and reference. The space is initially allocated when node is created and released when node is removed.

VXML supports VStorage transactions (catalog methods 'Begin', 'Commit' and 'RollBack').

When VXML storage is opened the transaction is started by default.

3. Extended XQL

VXQL supports XQL queries in 'SelectNodes' and 'SelectSingleNode' methods.
The only difference between these methods is that 'SelectSingleNode' will always select only one (first) node even if more than one node matches search criteria.
The query result depends on current node type:

- Node type '**catalog**'
  The result depends on the specified XQL prefix (1st character in the XQL expression):
  - '**$**' - the search will return *only catalog nodes*; documents are out of search criteria scope.
  - '**#**' - the search will return *only document nodes*; the document elements will also be analyzed for matching criteria if specified.

- Otherwise - the search will return *only document nodes*; only document names and attributes will be analyzed for matching criteria; the document elements will be ignored.

- Node type '**document**' or '**element**'
  Returns one or more nodes of 'element' type matching search criteria if the specific node type is not defined.
  Returns the node collection of the specified type if requested ('text', 'comment', 'content') matching the search criteria for 'element' nodes.
- Other node types: *XQL is not supported*.

3.1 vXQL syntax

<prefix>[<scope>]<context><[predicate 1][operation][predicate 2]…[operand][predicate n]…>

3.1.1 Prefix

Prefix defines search scope (for 'catalog' nodes only, otherwise shall be empty)

- '**$**' - the search will return *only catalog nodes*; documents are out of search criteria scope.
- '**#**' - the search will return *only document nodes*; the document elements will also be analyzed for matching criteria if specified.

3.1.2 Context

Context expression defines the scope for query. Context may contain special directives as well as node name pattern:

/// or //	- search recursively all tree starting from the root node ('element' or 'catalog';
.//	- search recursively all subtree starting from the current node;
..	- search subtree starting from the parent node;
.	- search subtree starting from the current node;
/	- search subtree starting from the root node;
first()	- select first node;
last()	- select last node;
content()	- select 'content' nodes;
name	- node name pattern to select (may contain '*' and '?' characters);

3.1.3 Predicate

Predicate defines filtering criteria in brackets: '[' and ']' or '{' and '}' (they are equal and could be used in multi-level predicates for better visibility.
Predicates could be multi-level. Predicates on the same level are connected by Boolean operation '&'(*and*) or '|'(*or*). If operand is missing, then assuming '&'.

Predicate can be one of the following:

1) **Node name**

   <name>

   Predicate is true if *child* node with the specified name exists.

   Node name can be a pattern including '*' and '?'.

   Example:

   ['name']

   {'family*'}

   ['a?dr*']

   ['*']

2) **Node number**

   <number>

   Predicate is true only if node with the specified number exists in the child nodes list for XPath.

   The numbering is starting from 1.

   Note: the number applies to the initial node list, not nodes in the query result.

   Example:

   [5]

   [19]

3) **Node tag(s)**

   [#tag1{#tag2}…{#tagN}]

   Predicate is true if at least one node tag exists. One or more tags separated by '#' could be specified.

   Tag values could be a pattern including '?' and '*' characters and are *not* case sensitive.

   Example:

   [#mytag]

   [#t?1#name*]

4) **Attribute name**

   @<name>

   Predicate is true if node attribute with the specified name exists.

   Attribute name can be a pattern including '*' and '?'.

5) Compare child node value

The node value is always treated as string.

<name><op><value>

Where:

<name> - child node name;

<op> - comparison operation, one of:

= equal, case sensitive;

!= not equal, case sensitive;

=$ equal, not case sensitive;

!=$ not equal, not case sensitive;

<value> - value for comparison, wildcards '*' and '?' are allowed.

Example:

Node1='*xt'

mynode !=$ 'firtst???'

6) Compare attribute value

The attribute value can be either string or numeric.

The string value must be quoted.

The numeric value must not be quoted and not contain alpha characters, otherwise error will be considered.

@<name><op><value>

Where:

<name> - attribute name;

<op> - comparison operation, one of:

= equal, numeric or string case sensitive;

!= not equal, numeric or string case sensitive;

=$ equal, string not case sensitive;

!=$ not equal, string not case sensitive;

> more than (numeric only);

< less than (numeric only);

>= equal or more than (numeric only);

<= equal, or less than (numeric only);

mailto: ivvvsx@gmail.com

<value> - value for comparison, wildcards '*' and '?' are allowed for string values.
Example:
atr1= 55
atr2 <= 17
myattr !=$ 'chapter ??'
name =$ '?nic*'

3.2 vXQL sample

node0005	Select child node 'node0005'.
node000*	Select all child nodes with the name starting with 'node000'.
node???5	Select all child nodes with the name starting with 'node' and ending with '5' and 3 any chars between.
*	Select all child element nodes.
//*	Select all element nodes in the document.
..	Select parent element node.
[1]	Select first child node of the current node.
[5]	Select fifth child node of the current node.
C_node[5]	Select fifth child node of the current node if the name is 'C_node'.
C_*[5]	Select fifth child node of the current node if the name starting with 'C_*'.
L*[first()]	Select first child node with the name starting with 'L'.
*[last()]	Select last child node.
*/*05	Select all grand-child nodes with the name ending with '05'.
*/*05[1]	Select first grand-child node if its name is ending with '05'.
*/*05[2]	Select second grand-child node if its name is ending with '05'.
/*[5]/*[3]	Select 3rd child of the 5th child of the root element node (grandchild of the root node).
v_0005//*	Select all descendant nodes of the child node with the 'v_0005' name.
v_0005//*[5]	Select 5th descendant node of the child node with the 'v_0005' name (look through all levels).
//v03_0005	Select all nodes in the document with 'v03_0005' name.

//v01_0005/v02_0003	Select all nodes with ‘v02_0003’ which are child of ‘v01_0005’ nodes through all document.
./	Select current node.
.	Select current node.
*5[@atr01='A01_06']	Select all child nodes with the name ending with ’5’ if node has attribute ‘atr01’ with the value ‘A01_06’.
*5[@atr01='A01*7']	Select all child nodes with the name ending with ’5’ if node has attribute ‘atr01’ which value matches pattern ‘A01*7’.
*[5][@atr07]	Select 5th child node if it has attribute with the name ‘atr07’.
[5][@atr01='A01']	Select 5th child node if it has attribute ‘atr01’ and its value matches case-sensitive pattern ‘A01*’.
[5][@atr01=$'a01']	Select 5th child node if it has attribute ‘atr01’ and its value matches non-case-sensitive pattern ‘a01*’.
l01_0005[l02_0003]	Select child node named ‘l01_0005’ if it has child node (grandchild of the current) named ‘l02_0003’.
*[1] | [5]	Select 1st and 5th child nodes.
*[[1]|[5]] & [[@atr5='A01_0005'] & [@atr3=’A01_0003']]	Select 1st or 5th child nodes if they have both ‘atr3’ and ‘atr5’ attributes with the specified values.
*{ { [1] | [5] } | [[@atr1=50] & [node2 = 99] }	Select 1st or 5th child nodes or any child node that has ‘atr1’ attribute with the numeric value ‘50’ or child element node (grandchild of the current) with the numeric value ‘99’.

4. Data portability

Each VXML document or element within the document cold be exported (‘checked out’) to the portable file format that can be imported (‘checked in’) to the same or another VXML database using ‘CheckOut’ method.

4.1 Check Out node

The ‘CheckOut’ method will perform the following actions:

1) Assign GUID to the current node. If node already has GUID assigned then existing value will be used.
2) Create portable file containing the full tree of the node. The file name is:

<GUID>.vsnp
Where:
<GUID> - GUID value assigned to this node;
vsnp – file extension

3) Make the current node and all subtree read-only

4.2 Export node

You can also use ‘Snap’ method to create portable file.

It is similar to ‘CheckOut’ method but remains the node and all subtree writable.

4.3 Undo Check Out node

‘UndoCheckOut’ method allows make the node and all subtree writable after checking out. But it retains the GUID value of node unchanged.

This method is useful if you need to cancel ‘CheckOut’ operation for node.

4.4 Check In node

You can check in portable file into the VXML database. The following actions will apply:

1) If GUID from the portable file already exists in the VXML database then node associated with this GUID will be used as place for checking in, no matter which node is current.
2) If GUID doesn’t exist in the VXML database then it will be assigned to node checked in.
3) If GUID exists then this node will be replaced by the portable file content otherwise new child node and subtree will be created as child of the current node.
4) The node will become writeable (if it was previously checked out).

III. API

The **VXML.dll** is virtual XML storage component which is based on Virtual Storage platform.

It provides such capabilities as:

- Manage catalog structure for XML documents;
- Manage XML documents and nodes;
- Support XQL queries for catalog nodes and documents;
- Data portability between multiple VXML storages;
- Support transactions via Virtual Storage transaction mechanism;
- Binary content management.

Programming language – C#

Framework - .NET Framework 4.5

Dependencies – Vstorage.dll

1. Public Constants and Business Rules

1.1 public static class DEFX

VXML node types numeric values of the 'VXmlNode.NodeTypeCode' field. The corresponding string representation can be obtained using 'VXmlNode.NodeType' field.

Constant	Description
public const short NODE_TYPE_UNDEFINED = 0	Undefined
public const short NODE_TYPE_CATALOG = 1	'catalog' node type
public const short NODE_TYPE_DOCUMENT = 2	'document' node type
public const short NODE_TYPE_ELEMENT = 3	'element' node type
public const short NODE_TYPE_CONTENT = 4	'content' node type
public const short NODE_TYPE_REFERENCE = 5	'reference' node type
public const short NODE_TYPE_ATTRIBUTE = 100	'attribute' node type
public const short NODE_TYPE_COMMENT = 101	'comment' node type
public const short NODE_TYPE_TEXT = 102	'text' node type
public const short NODE_TYPE_TAG = 103	'tag' node type

Other constants:

Constant	Description
public static string GET_NODETYPE(short type)	Returns string representation of the node type code.
public const string KEY_LOAD_XML = "loadxml-path.vs.default"	VStorage key for load xml directory
public const string KEY_SNAP = "snap-path.vs.default"	VStorage key for checkout/snap directory.
public const string PREFIX_ATTRIBUTE="%"	VSObject field prefix for attribute name
public const string PREFIX_COMMENT="^"	VSObject field prefix for comment node name

public const string PREFIX_TEXT="&"	VSObject field prefix for text node name
public const string NON_START_PATTERN	String containing additional characters allowed in the node name
public const string START_PATTERN	String containing characters allowed in the node name 1st symbol
public const string NON_START_PATTERN	String containing additional characters allowed in the node name
public const string XML_CONTENT_SPACE_NAME="vxmlcont"	VStorage space name for binary content
public const string XML_INDEX_SPACE_NAME="vxmlindx"	VStorage space name for indexing
public const string XML_SNAP_FILE_TYPE = "vsnp"	Portable file type (extension) for Snap/CheckOut
public const string XML_SPACE_NAME="vxmlbase"	VStorage space name for XML data

Business rules:

Constant	Description
public static bool BR_CAN_HAVE_REFERENCE(short type)	Returns true if node type can have reference.
public static bool BR_CHECKIN_IS_VALID_TYPE(short parent_type)	Returns true if 'CheckIn' is allowed for node type.
public static bool BR_CHECKOUT_IS_VALID_TYPE(short parent_type)	Returns true if 'CheckOut' or 'Snap' is allowed for node type.
public static bool BR_CHILD_IS_VALID_TYPE(short parent_type, short type)	Returns true if node of type 'parent_type' can create child node of type 'type'.
public static short[] BR_CHILD_VALID_TYPE_CODES(short type)	Returns array of the valid types for child node.
public static string[] BR_CREATE_VALID_TYPES(short type)	Returns string array of node types that can be created as child nodes for node type.
public static bool BR_INSERT_IS_VALID_TYPE(short target_type, short type)	Returns true if node of type 'type' can be inserted before the node of type'target_type'.
public static bool BR_NODE_NAME_VALID(string name)	Returns true if specified string is valid for node name.
public static bool BR_NODE_NEED_NAME(short type)	Returns true if node name is user-defined.
public static bool BR_NODE_REFERENCE(short type)	Returns true if reference can be created for node type.
public static bool BR_NODE_RENAME(short type)	Returns true if node of this type can be renamed.
public static bool BR_XML_IS_VALID_TYPE(short parent_type)	Returns true if load or save XML is allowed for node type.

2. Classes

2.1 public class VXmlNode

Root VXML node class.

2.1.1 Properties

Property	Description
public VXmlNodeCollection AllChildNodes	The collection of all child nodes of all types, including 'attribute' nodes.
public VXmlAttributeCollection Attributes	The collection of child 'attribute' nodes.
public bool CheckedOut	True if current node is checked out.
public bool CheckedOutTree	True if current node or any node in the subtree is checked out.
public VXmlNodeCollection ChildNodes	The collection of child 'element' nodes.
public VXmlCommentCollection CommentNodes	The collection of child 'comment' nodes.

public VXmlContentCollection ContentNodes	The collection of child ‘content’ nodes.
public VSpace ContentSpace	Vrtual Strorage space for binary content.
public VXmlNode FirstChild	First ‘element’ node.
public string GUID	CheckOut/Snap GUID of the current node; empty if node has never been checked out.
public bool HasAttributes	True if node has at least one attribute.
public bool HasChildNodes	True if node has at least one child node (except attribute).
public long Id	Unique node Id within storage numeric representation.
public long ID	Unique node Id within storage string representation.
public VXmlNode LastChild	Last ‘element’ node.
public string Name	Node name.
public VXmlNode Next	Next node of any type.
public VXmlNode NextSibling	Next ‘element’ node.
public VSpace NodeSpace	Virtual Storage space for XML data.
public string NodeType	Node type string representation.
public short NodeTypeCode	Node type numeric representation.
public VXmlNode Owner	Owner node for current node (‘document’ or ‘catalog’).
public VXmlDocument OwnerDocument	Owner document (for all node types except ‘document’ or ‘catalog’).
public VXmlNode ParentNode	Parent node of the current node.
public VXmlNode Previous	Previous node of any type.
public VXmlNode PreviousSibling	Previous ‘element’ node.
public VXmlTagCollection Tags	The collection of child ‘tag’ nodes.
public string Text	Concatenation of all text nodes values.
public VXmlTextCollection TextNodes	The collection of child ‘text’ nodes.
public string Value	Node value.
public string Xml	String representation of the node and subtree.

2.1.2 Constructors

No public constructors.

2.1.3 Methods

Name	Parameter(s)	Description
public VXmlNode AppendChild(VXmlNode node)	**node** – node to append	Append existing node as child.
public VXmlNode CheckIn(string path)	**path** – full path to ‘vsnp’ file	Check in XML node from the portable file format.
public string CheckOut(string path)	**path** – full path to directory where file shall be created (fine name must not be included)	Check out node (export to the portable format file).
public VXmlNode Clone()		Create clone of the node including all subtree.
public VXmlNode CloneNode(bool deep)	**deep** – ‘true’ to clone full tree; otherwise false (default)	Create clone of the node itself of all tree depending on parameter.

public VXmlAttribute CreateAttribute(string name, string value)	**name** – attribute name **value** – attribute value	Create new attribute node.
public VXmlAttribute CreateAttributeAt(string xpath, string name, string value)	**xpath** – node location **name** – attribute name **value** – attribute value	Create new attribute node at the specified XPath.
public VXmlComment CreateComment(string text)	**text** – comment text	Create new comment node.
public VXmlComment CreateCommentAt(string xpath, string text)	**xpath** – node location **text** – comment text	Create new comment node at the specified XPath.
public VXmlContent CreateContent(string title, string filename)	**title** – content title **filename** – full path to the binary content file	Create new content node.
public VXmlContent CreateContentAt(string xpath, string title, string filename)	**xpath** – node location **title** – content title **filename** – full path to the binary content file	Create new content node at the specified XPath.
public VXmlElement CreateElement(string name, string value)	**name** – node name **value** – node value	Create new element node.
public VXmlElement CreateElementAt(string xpath, string name, string value)	**xpath** – node location **name** – node name **value** – node value	Create new element node at the specified XPath.
public VXmlTag CreateTag(string text)	**text** – tag value. I tag with the specified value already exists, the new node will not be created, existing node will be returned.	Create new tag node.
public VXmlText CreateTextNode(string text)	**text** – text value	Create new text node.
public VXmlText CreateTextNodeAt(string xpath, string text)	**xpath** – node location **text** – text value	Create new text node at the specified XPath.
public string GetAttribute(string name)	**name** – attribute name	Get attribute value.
public string GetAttributeAt(string xpath, string name)	**xpath** – node location **name** – attribute name	Get attribute value at the specified XPath.
public VXmlAttribute GetAttributeNode(string name)	**name** – attribute name	Get attribute node.
public VXmlElement GetChildElement(string name)	**name** – node name	Get child element node by name.
public VXmlComment GetCommentNode(string name)	**name** – node name	Get comment node by name node.
public VXmlNode GetFirstNode(short type)	**type** – node type	Get first child node of the specified type.
public VXmlNode GetLastNode(short type)	**type** – node type	Get last child node of the specified type.
public VXmlNode GetNode(long id)	**id** – node unique id in the storage	Get node by unique Id. This node doesn't have be a child node.
public VXmlText GetTagNode(string name)	**name** – node name	Get tag node by node name.
public VXmlText GetTextNode(string name)	**name** – node name	Get text node by node name.
public VXmlNode InsertAfter(VXmlNode node, VXmlNode after)	**node** – node to insert **after** – child node to insert after	Insert node after the specified child node.

public VXmlNode InsertBefore(VXmlNode node, VXmlNode before)	**node** – node to insert **before** – child node to insert before	Insert node before the specified child node.
public string Load(string file, VXmlNode parent = null)	**file** – full path to the XML file **parent** – node to which new node will be added as child. If null then current node will be parent.	Create XML node(s) from the XML representation in text file. All tree will be created if child nodes present.
public string LoadXml(string xmlstring, string name, VXmlNode parent = null)	**xmlstring** – XML string representation **name** – name for cache; using the name is useful to improve performance if many nodes with the same name and structure created. If name matches cache then previously parsed result will be used. **parent** – node to which new node will be added as child. If null then current node will be parent.	Create XML node(s) from the XML string representation. All tree will be created if child nodes present.
public VXmlNode PrependChild(VXmlNode node)	**node** – node to prepend	Similar to 'AppendChild' but node will be added as first child.
public void Remove()		Remove the current node and all subtree.
public void RemoveAll()		Remove all child nodes and attributes of the current node.
public void RemoveAllAt(string xpath)	**xpath** – node location	Remove all child nodes and attributes at the specified XPath.
public void RemoveAllAttributes()		Remove all attributes of the current node.
public void RemoveAttribute(string name)	**name** – attribute name	Remove attribute of the current node by name.
public void RemoveChild(VXmlNode node)	**node** – child node object	Remove child node.
public void RemoveNodeAt(string xpath)	**xpath** – node location	Remove node at specified XPath.
public void RemoveText(string name)	**name** – node name	Remove text node by name.
public void RemoveTag(string tag)	**tag** – tag value	Remove tag node by value.
public void RemoveTagByName(string name)	**name** – node name	Remove tag node by name.
public void ReplaceChild(VXmlNode newChild, VXmlNode oldChild)	**oldchild** – node to be replaced **newchild** – node for replacement	Replace child node by another node.
public string SaveXml(string fname, bool content = true)	**fname** – full path to the XML file **content** – true: save binary content	Save node and subtree to file in XML format.
public VXmlAttributeCollection SelectAttributes(string xpath, string name)	**xpath** – node location **name** – attribute name or pattern. All attributes by default.	Select node attributes using XPath expression for node. If more than one node matches XPATH expression then 1st node in the search result will be used.
public VXmlNodeCollection SelectNodes(string xpath)	**xpath** – node search expression	Select one or more nodes using XPath expression.

public VXmlNode SelectSingleNode(string xpath)	**xpath** – node search expression	Select one or more nodes using XPath expression. If more than one node is in the query result then only first will be returned.
public void SetAttribute(string name, string value)	**name** – attribute name **value** – attribute value	Create node attribute or update value of the existing attribute.
public void SetAttributeAt(string xpath, string name, string value)	**xpath** – node location **name** – attribute name **value** – attribute value	Create node attribute or update value of the existing attribute attributes at the specified XPath.
public string Snap(string path = "")	**path** – full path to directory where file shall be created (fine name must not be included)	Similar to 'CheckOut' but doesn't lock node(s).
public bool UndoCheckOut()		Turns off read-only mode for checked out tree.

2.2 `public class VXmlContent : VXmlNode`

Binary content management (node type 'content').

2.2.1 Properties

Property	Description
public byte[] ContentBytes	Binary content representation as byte array.
public string ContentString	Binary content representation as byte string.
public string filename	The file name if content is uploaded from file.
public string fileref	The file reference (URL) if content has been saved (generated by SaveXml).
public long Length	Content length (bytes).
public string path	The file path if content is uploaded from file.
public long size	The file size in bytes if content is uploaded from file.
public string title	Content title.

2.2.2 Constructors

No public constructors.

2.2.3 Methods

Name	Parameter(s)	Description
public void Download(string path)	**filename** – full path and file name to save. If empty then saved file name will be used. If saved name is empty then content will be saved to the current directory with 'NONAME.content' name.	Download binary content to file.

public void Upload(string file, bool setattr)	**filename** – full path to the file to upload **setattr** – if true (default) fields 'filename' and 'path' will be set	Upload binary content from file.

2.3 public class VXmlReference : VXmlNode

Node reference (node type 'reference').

2.3.1 Properties

Property	Description
public long ReferenceId	Unique Id of the original node.
public VXmlNode ReferenceNode	Original node object.

2.3.2 Constructors

No public constructors.

2.3.3 Methods

N/A

2.4 public class VXmlDocument : VXmlNode

VXML document (node type 'document').

2.4.1 Properties

Property	Description
public VXmlNode DocumentElement	Root element of the document.

2.4.2 Constructors

No public constructors.

2.4.3 Methods

Name	Parameter(s)	Description
public VXmlDocument Clone(string new_name)	**new_name** – name for new document	Create clone of the document. New name will be assigned to new document.
public string Load(string file, VXmlNode parent)	**file** – full path to the XML file **parent** – node to which new node will be added as child. If null then current node will be parent.	Create XML node(s) from the XML representation in text file. All tree will be created if child nodes present.

public string LoadXml(string xmlstring, string name, VXmlNode parent)	**xmlstring** – XML string representation **name** – name for cache; using the name is useful to improve performance if many nodes with the same name and structure created. If name matches cache then previously parsed result will be used. **parent** – node to which new node will be added as child. If null then current node will be parent.	Create XML node(s) from the XML string representation. All tree will be created if child nodes present.

2.5 `public class VXmlCatalog : VXmlDocument`

VXML catalog node (node type 'catalog').

2.5.1 Properties

Property	Description
public VXmlCatalogCollection ChildNodes	List of child catalog nodes.
public VXmlDocumentCollection Documents	List of document nodes for catalog node.
public bool HasChildNodes	True if node has at least one child catalog node (or reference to catalog node).
public VXmlCatalog OwnerCatalog	Owner catalog node.
public VXmlReferenceCollection References	List of reference nodes for catalog node.
public VSEngine Storage	Virtual Storage `VSEngine` object.
public title Storage	Catalog node title

2.5.2 Constructors

Constructor	Description
public VXmlCatalog()	This constructor can be used if you need to initialize the ne new storage ('InitializeStorage' method).
public VXmlCatalog(string root_path)	Create root catalog object only. '**root_path**' parameter specifies path to Virtual Storage root catalog. If storage doesn't exist then it will be created with the default parameters (XML space – 5 Mb, XML space auto extension – 5 Mb; content space – 5 Mb, content space auto extension – 15 Mb).

2.5.3 Methods

Name	Parameter(s)	Description
public void Begin()		Begin transaction.
public void Close()		
public void Commit()		Commit transaction.

public VXmlCatalog CreateCatalog(string name, string value)	**name** – catalog node name **value** – catalog name value (optional)	Create new 'catalog' node.
public VXmlDocument CreateDocument(string name)	**document** – catalog node name	Create new document.
public VXmlReference CreateReference(VXmlNode n)	**n** – node for reference	Create new node reference.
public VXmlCatalog GetChildCatalog(string name)	**name** – catalog node name	Get child catalog node by name.
public VXmlDocument GetChildDocument(string name)	**name** – document name	Get child document node by name.
public void InitializeStorage(string root_path, int size, int ext = 0, int content_size = 0, int content_ext = 0)	**root_path** – root storage directory in the file system (must exist) **size** – XML space initial size (Mb) **ext** – XML space auto extension (Mb), default - 0 **content_size** – content space initial size (Mb), default - 0 **content_ext** – content space auto extension (Mb), default - 0	Create and initialize new storage.
public void RemoveAllReferences()		Remove all references for this catalog node.
public void RemoveCatalog(VXmlCatalog cat)	**cat** – catalog node	Remove child catalog node.
public void RemoveDocument(VXmlDocument doc)	**doc** – document	Remove child document.
public void RemoveReference(VXmlReference r)	**r** – reference node	Remove child reference node.
public void RollBack()		Rollback transaction.

2.6 `public class VXmlException : Exception`

Inherited from: Exception

2.6.1 Constructors

Constructor	Parameter(s)	Description
public VXmlException(int code, string message_ext)	**code** – error code **message_ext** – message to append to the default system message (empty by default).	VXML exception object. Any exception thrown has its own code and message.

2.6.2 Properties

Property	Description
public int ErrorCode	Error code for exception
public string Message	Full error message for exception

2.6.3 Error codes and messages

Constant	Message	

public int E0001_CONTENT_FILE_NOT_FOUND_CODE = 1	Content file not found	The specified file is not found in the 'CreataContent' method
public int E0002_NODE_IS_UP_TREE_CODE = 2	Node is up in the tree	Node for append/insert/replace is parent for current node
public int E0003_NODE_IS_FROM_DIFFERENT_DOC_CODE = 3	Node is up from different document	Node for append/insert/replace/remove is not in the current document
public int E0004_INVALID_NODE_TYPE_CODE = 4	Invalid node type	Node type is not applicable in the current operation
public int E0005_NOT_A_CHILD_NODE_CODE = 5	Not a child node	The specified node is not a child node in replace/remove/insert operation
public int E0006_CATALOG_INVALID_OP_CODE = 6	Invalid operation, applicable for root catalog element only	Transaction management methods or 'Close' method is invoked not by the root catalog node
public int E0007_OLD_EQUAL_NEW_CODE = 7	Operands are the same node	Node in insert/replace operations are the same
public int E0008_INVALID_CHAR_CODE = 8	Invalid character in the node name	Invalid character in the node name
public int E0009_INVALID_TYPE_CODE = 9	Invalid node type	Node type is invalid for operation (create node, check out)
public int E0010_ROOT_EXISTS_CODE = 10	CreateNode: root node already exists	Attempt to create root element for document when it already exists
public int E0011_XML_CREATE_INVALID_TYPE_CODE = 11	Create from template - node type is not element	The nNode type initiated 'Load' or 'LoadXml' method is not 'element'
public int E0012_PATH_NOT_FOUND_CODE = 12	Path is not found	File path is not found in CheckOut operation
public int E0013_ALREADY_CHECKED_OUT_CODE = 13	"Current node, parent or child node is checked out	Attempt of any operation affected checked out node
public int E0014_UNSUPPORTED_CHECKOUT_VERSION_CODE = 14	Unsupported CheckOut format version	'CheckIn' found that portable file format is not compatible with the current VXML version
public int E0015_CHECKIN_ERROR_CODE = 15	CheckIn error:	Error during 'CheckIn', the particular details are provided in the message
public int E0016_XQL_ERROR_CODE = 16	XQL error:	XQL format error in 'SelectNodes' or 'SelectSingleNode', the particular details are provided in the message
public int E0017_DOC_EXISTS_CODE = 17	Node with the specified name already exists	Duplicate name when trying to rename node attribute
public int E0018_DOC_NAME_MISSING_CODE = 18	Document name is missing	New document name is not specified when cheating document clone
public int E0019_SPACE_MISSING_CODE = 19	VXML Node space is not found	Virtual Storage space for xml data is not found when initiating root catalog object
public int E0020_XML_PARSE_ERROR_CODE = 20	VXml Parser - parse error	Unexpected XML parsing error in 'Load' or 'LoadXml' methods
public int E0021_INVALID_NODE_FIELD_CODE = 21	Invalid node field (internal error)	Possible reason: physical storage structure is damaged

public int E0022_NOT_VSXML_NODE_SPACE_CODE = 22	Space owner is not VSXML or undefined	The specified VXML storage is actually designed to another application or damaged
public int E0023_NOT_EMPTY_UNDEFINED_SPACE_CODE = 23	Space owner is undefined but space is not empty	The specified VXML storage is ready to be initialized for VXML but actually is not empty
public int E0024_ONE_NODE_MUST_BE_SELECTED_CODE = 24	One node must be selected	Attempt to perform 'At' operation but XQL returned mode than one node for the specified XPath
public int E0025_XML_FILE_ERROR_CODE = 25	VXml Parser - file read error	Unexpected error when reading XML file in 'Load' method

2.6.4 Methods

Name	Parameter(s)	Description
public static string GetMessage(int code)	**code** – error code	Static method, returns the basic error message for error code.

2.7 `public class VXmlDummy`

Parent class for internal node types and their collections.

2.7.1 Properties

Property	Description
public long Id	Always 0.
public string ID	Always "N/A".

2.7.2 Constructors

No public constructors.

2.7.3 Methods

No public methods.

2.8 `public class VXmlAttribute : VXmlDummy`

Class for 'attribute' node type.

2.8.1 Properties

Property	Description
public string Name	Attribute name.

public string NodeType	Node type string representation.
public short NodeTypeCode	Node type numeric representation.
public string Value	Attribute value.

2.8.2 Constructors

No public constructors.

2.8.3 Methods

No public methods.

2.9 `public class VXmlComment : VXmlDummy`

Class for 'comment' node type.

2.9.1 Properties

Property	Description
public string Name	Node name, system-generated.
public string NodeType	Node type string representation.
public short NodeTypeCode	Node type numeric representation.
public string Value	Comment value.

2.9.2 Constructors

No public constructors.

2.9.3 Methods

No public methods.

2.10 `public class VXmlText : VXmlDummy`

Class for 'text' node type.

2.10.1 Properties

Property	Description
public string Name	Node name, system-generated.
public string NodeType	Node type string representation.
public short NodeTypeCode	Node type numeric representation.

public string Value	Text value.

2.10.2 Constructors

No public constructors.

2.10.3 Methods

No public methods.

2.11 `public class VXmlTag : VXmlDummy`

Class for ‘tag’ node type.

2.11.1 Properties

Property	Description
public string Name	Node name, system-generated.
public string NodeType	Node type string representation.
public short NodeTypeCode	Node type numeric representation.
public string Value	Tag value.

2.11.2 Constructors

No public constructors.

2.11.3 Methods

No public methods.

2.12 `public class VXmlAttributeCollection : VXmlDummy`

Node attributes collection.

2.12.1 Properties

Property	Description
public int Count	The number of nodes in the collection.
public VXmlAttribute this[int index]	Attribute node by index.
public VXmlAttribute this[string name]	Attribute node by node name

2.12.2 Constructors

No public constructors.

2.12.3 Methods

No public methods.

2.13 `public class VXmlCommentCollection : VXmlDummy`

Node comments collection.

2.13.1 Properties

Property	Description
public int Count	The number of nodes in the collection.
public VXmlComment this[int index]	Comment node by index.
public VXmlComment this[string name]	Comment node by node name

2.13.2 Constructors

No public constructors.

2.13.3 Methods

No public methods.

2.14 `public class VXmlTextCollection : VXmlDummy`

Text nodes collection.

2.14.1 Properties

Property	Description
public int Count	The number of nodes in the collection.
public VXmlText this[int index]	Text node by index.
public VXmlText this[string name]	Text node by node name

2.14.2 Constructors

No public constructors.

2.14.3 Methods

No public methods.

2.15 `public class VXmlTagCollection : VXmlDummy`

Tag nodes collection.

2.15.1 Properties

Property	Description
public int Count	The number of nodes in the collection.
public VXmlText this[int index]	Tag node by index.
public VXmlText this[string name]	Tag node by node name

2.15.2 Constructors

No public constructors.

2.15.3 Methods

No public methods.

IV. VXML Administration Tool

Administration tool (VXmlExplorer) allows Virtual XML storage management via GUI.

1. **Main menu ('File')**

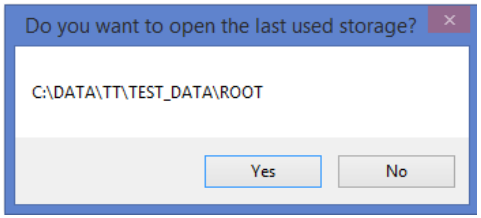
   The 'File' menu contains two kinds of items:

   1) VXML Storage management (open, close, create)
   2) Node-specific functions when the particular node is selected (also available in the context menu).

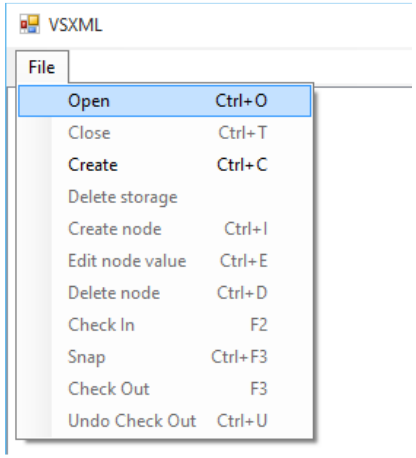
   1.1 Open VXML Storage

   Note: the storage must be created before opening.

   If any storages was opened previously then system will prompt the confirmation to open this storage:

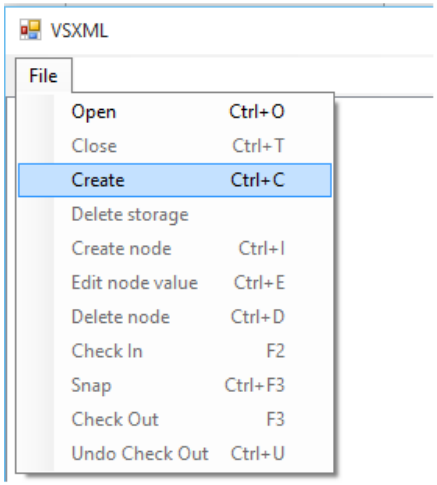


Otherwise you should use ‘Open’ menu and select the storage directory:

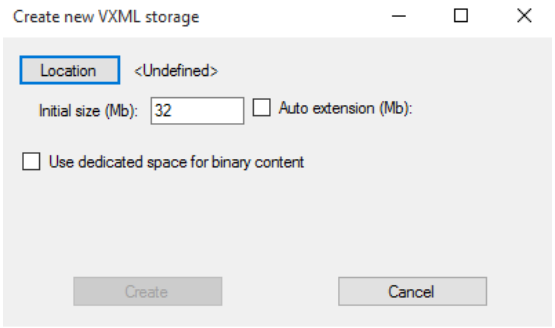


1.2 Create VXML Storage

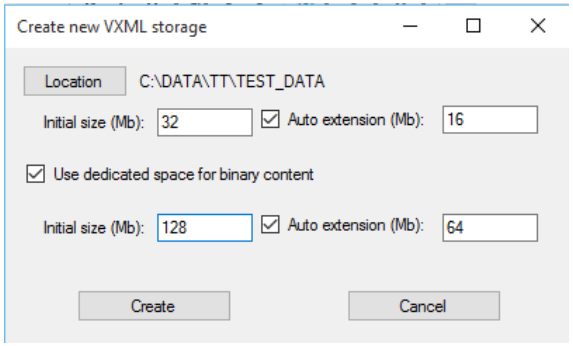
Use ‘Create’ menu to create new VXML storage:



New pop-up window will be opened to define the storage parameters:



'Location' is mandatory parameter, you shall specify the root directory for storage.
'Auto extension' option can be used if you want to extend XML storage dynamically if the current size is insufficient.
'Use dedicated storage for binary content' allows create separate VStorage space for binary content:



Initial size and extension (optional) should be specified for this space.

1.3 Close VXML Storage

Close storage. After the operation is complete all other functions except ‘Open’ will be disabled.

1.4 Delete VXML Storage

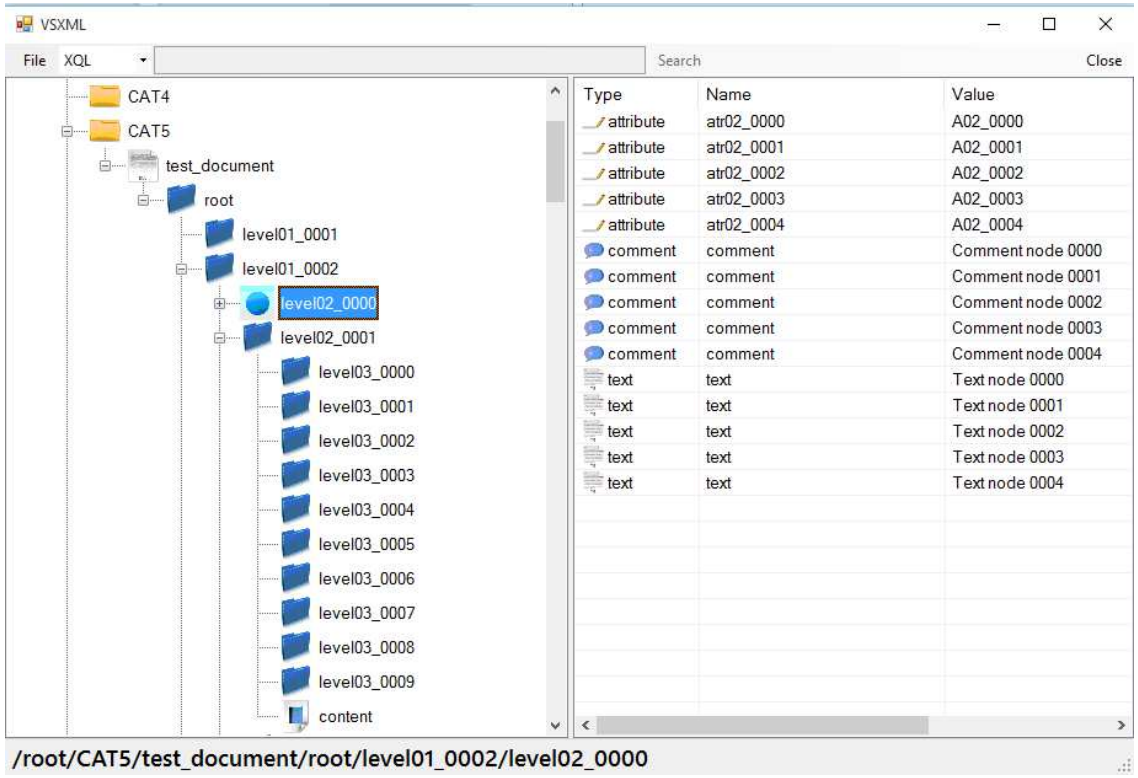
VXML Storage will be closed, XML data and binary content spaces physically deleted from the file system and Virtual Storage catalog.

Make sure you have backup copy of the storage before performing this action because roll back is not supported in this case.

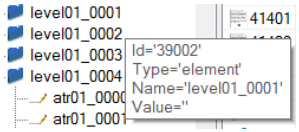
2. Storage navigation

The mail screen contains two panels:

- Tree view panel (left) – displays the XML tree in the storage. Node types included: catalog, document, element, content, reference.
- Detail (list) view panel (right) – displays the detail list of nodes for selected node in the tree view. Node types included: text, attribute, comment.



If you position the mouse cursor upon any node in the tree the tooltip text will show the node information:



Click on the node in the node tree will populate the list of all child nodes in the list view.

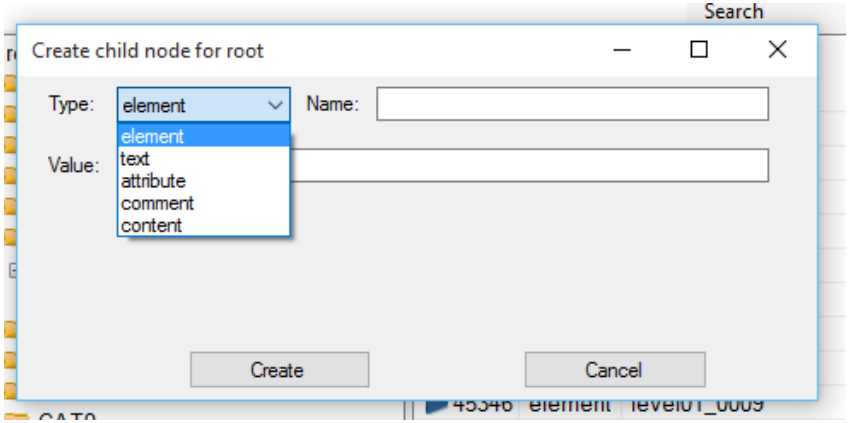
Doubleclick on the node in the list view will automatically select the node in the tree view and display all its children in the list view.

3. Context menu

The context is displayed by the mouse right click on the node in the tree view of list view.
The set of menu items may differ depending on view (tree or list), node type or node current state.

3.1 Create Node

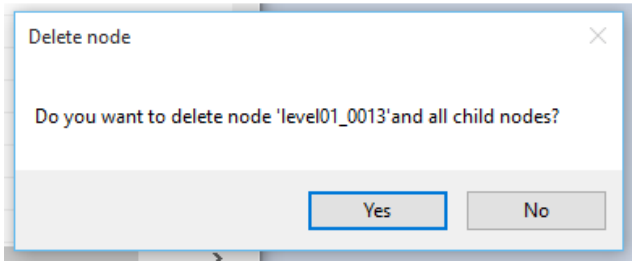
The pop-up window for node creation will be opened:



The list of node types for creation may differ depending on current node type.
Only allowed node types will be displayed.
New node will be created and added as child of the current node.

3.2 Delete Node

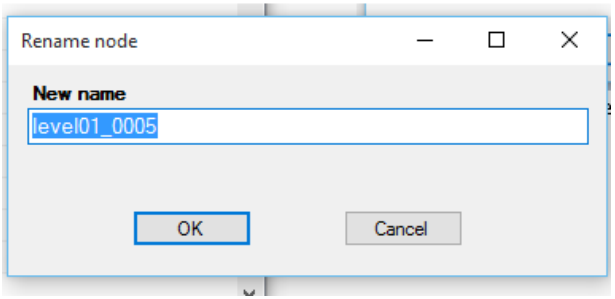
Confirmation pop-up window will be displayed:



Fter the confirmation this node and all child nodes will be deleted.

3.3 Rename Node

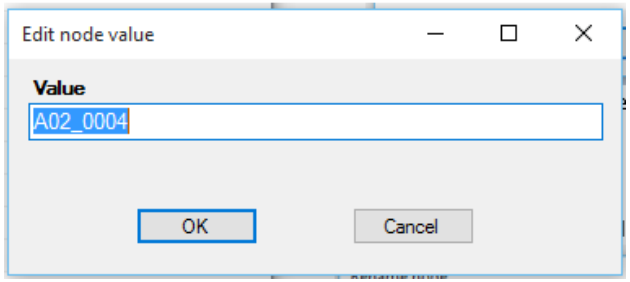
This operation is allowed only for nodes that can be renamed.
The pop-up window requesting the new node name will be displayed.



After renaming the node will be opened in the tree view.

3.4 Edit Value

The pop-up window requesting the new node value will be displayed.



The current node in the view will remain, displayed value changed.

3.5 Lookup original node

This function is available for ‘reference’ node type only (displayed with ‘~’ prefix.
The original node location will be opened in the tree view for this reference.

3.6 Check In

Check in node from the portable file, standard dialog window will be opened to specify the file location.
If GUID associated in the portable file doesn’t exist in this storage then XML data from the file will be added as a child of the current node. Otherwise the node identified by this GUID will be replaced (this can be not current node).
The checked in node will be opened in the tree view.

3.7 Check Out

Current node will be checked out to the portable file format (‘vsnp’).
Standard dialog window will be opened to specify the folder where this file will be saved.
The name of the file will be equal to GUID for this node (it will be assigned automatically if node have no GUID yet).
The node and all subtree will become read-only if operation is successful.

3.8 Snap

This operation is equal to ‘Check Out’ but only ‘vsnp’ file will be created, the node and all subtree will remain writable.

3.9 Undo CheckOut

Change state to writable for the node and all its subtree if it was checked out.

3.10 Copy node name to clipboard

Place the current node name to the clipboard. This could be useful to create XQL expressions.

3.11 Download Content

This operation is applicable only for ‘content’ node type.
The binary content will be saved to file, standard ‘Save’ dialog window will be displayed.

3.12 View XML

Display the current node and subtree as XML representation in the text box.
The binary content will not be included for ‘content’ nodes.

3.13 Save XML

Save the current node and subtree as XML representation in the text file.
The file name is equal to node name, the file type is ‘xml’.
The binary content will not be included for ‘content’ nodes.

3.14 Save XML wit content

Save the current node and subtree as XML representation in the text file.

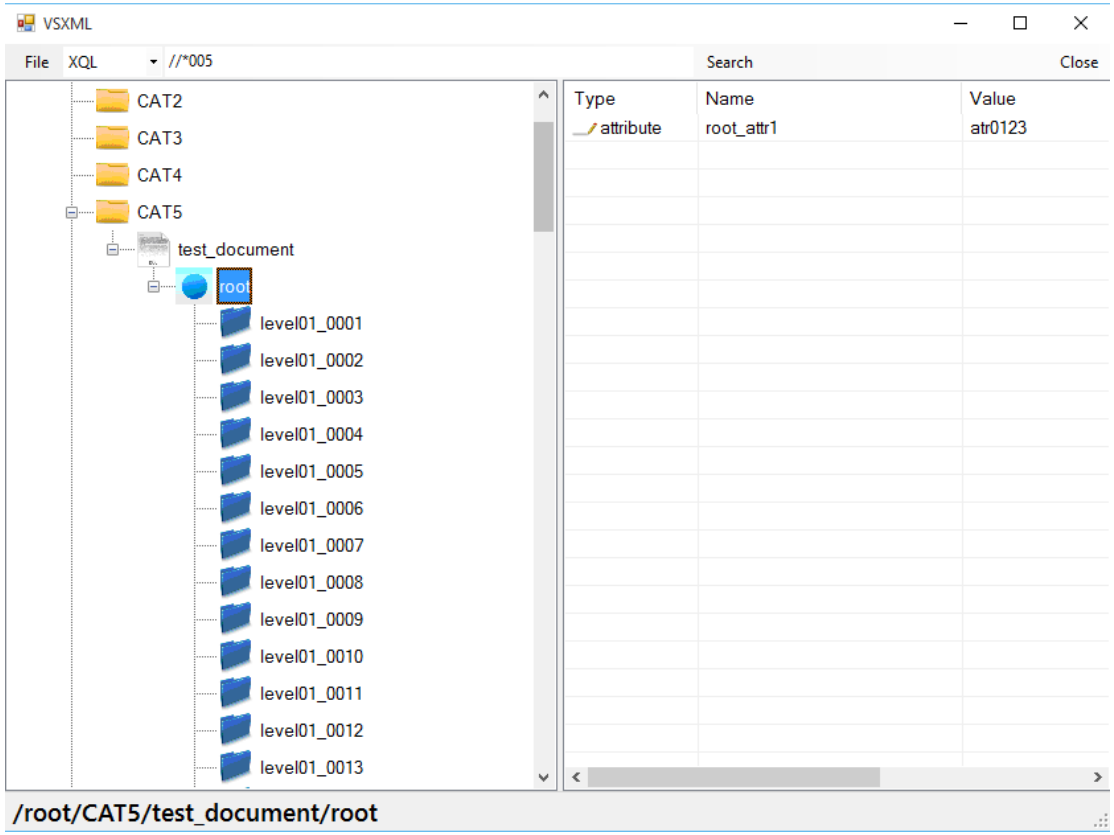
The file name is equal to node name, the file type is ‘xml’.
The binary content will be saved in ‘files’ directory, the corresponding references will be added to xml file as ‘fileref’ attribute of the ‘content’ node.

3.15 Load XML

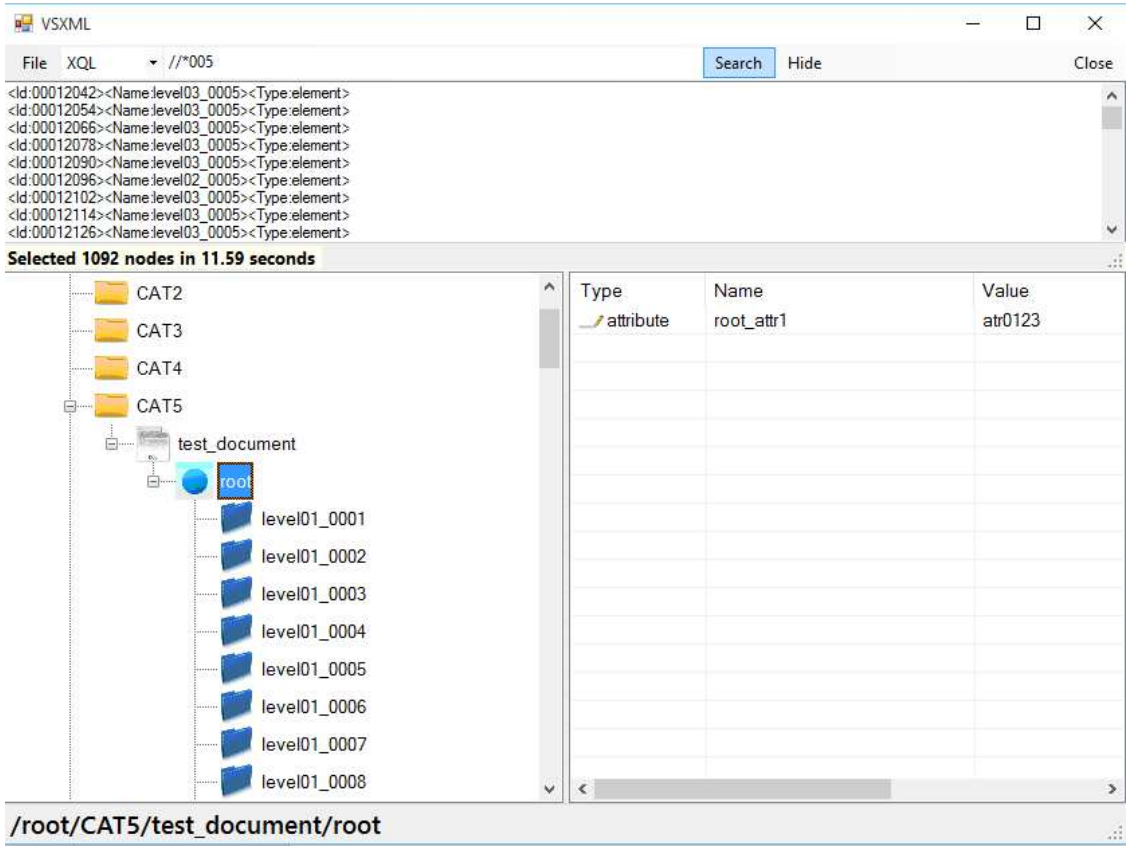
Load XML data from the XML file and append as a child of the current node.
The binary content will be loaded if XML file has ‘fileref’ attribute of the ‘content’ node.

4. **Execute XQL query**

You can enter XQL expression in the text box right of ‘File’ menu.
The search will be performed relatively to the current node.
See ‘3. Extended XQL’ chapter for search expression format.
Search starts by pressing ‘Enter’ key or clicking ‘Search’ button on the right of the text box:



After completion the search result panel will be displayed:



Doubleclick on the node in the search result list will open this node in the tree view.
'Hide' button will be visible after the query and can be used to hide search result list.

V. Code sample

1. Open VXML storage

```
ROOT = "";
if (path == "")
{
    ROOT = VSUILib.VSUICommonFunctions.SelectPath(DEFS.KEY_STORAGE_ROOT, "Select MXML storage path");
        if ((ROOT == VSUILib.VSUICommonFunctions.CANCELLED) | (ROOT == ""))
        {
            ROOT = "";
            return;
        }
}
else
    ROOT = path;

try
{
    cont = new VXmlCatalog(ROOT);
}
catch (VXmlException e)
{
    MessageBox.Show(e.Message, "VXML Error", MessageBoxButtons.OK);
    cont.Close();
    return;
}
cont.Commit();
```

2. CheckOut node

```
if (CONTEXT_XML_NODE.CheckedOut)
{
    MessageBox.Show("Node is already checked out", "Error", MessageBoxButtons.OK);
    return;
}

// Prompt directory
string st = VSUILib.VSUICommonFunctions.SelectPath(DEFX.KEY_SNAP, "Select target directory");
if (st != "")
{
    bool state = true;

    if (lck)
        tsNode.Text = "Checking out " + CONTEXT_XML_NODE.Name + " ...";
```

```
    else
        tsNode.Text = "Snapping " + CONTEXT_XML_NODE.Name + " ...";

    Application.DoEvents();

    cont.Begin();
    try
    {
        CONTEXT_XML_NODE.CheckOut(st, lck);
    }
    catch (VXmlException e1)
    {
        MessageBox.Show(e1.Message, "Error", MessageBoxButtons.OK);
        state = false;
    }
    cont.Commit();
    if (state)
    {
        SELECT_Node();
        string op = lck ? "Check Out" : "Snap";
        MessageBox.Show(op + " completed, GUID=" + CONTEXT_XML_NODE.GUID, "Successful", MessageBoxButtons.OK);
    }
```

3. Display child nodes in the 'TreeView' control

```
VXmlNode node;
TreeNodeCollection tc;
Cursor.Current = Cursors.WaitCursor;

tvCat.BeginUpdate();

if (CONTEXT_TREE_NODE == null)
{
    node = cont;
    CONTEXT_XML_NODE = cont;
    CONTEXT_TREE_NODE = tvCat.Nodes.Add(node.ID, UTIL_PrepareName(node));
    CONTEXT_TREE_NODE.ImageKey = node.NodeType;
    ACTION_SetTreeNodeName(CONTEXT_TREE_NODE, node);
    tc = CONTEXT_TREE_NODE.Nodes;
    tvCat.SelectedNode = CONTEXT_TREE_NODE;
}
```

```
            else
            {
                tc = CONTEXT_TREE_NODE.Nodes;
                node = CONTEXT_XML_NODE;
            }
            tc.Clear();

            if (node.NodeTypeCode != DEFX.NODE_TYPE_ATTRIBUTE)
            {
                    VXmlNode c = node.FirstAttribute;
                    while (c != null)
                    {
                        TreeNode tn = tc.Add(c.ID, UTIL_PrepareName(c));
                        tn.ImageKey = c.NodeType;
                        ACTION_SetTreeNodeName(tn, c);
                        c = c.Next;
                    }


                    c = node.First;
                    while (c != null)
                    {
                        TreeNode tn = tc.Add(c.ID, UTIL_PrepareName(c));
                        tn.ImageKey = (c.NodeTypeCode == DEFX.NODE_TYPE_REFERENCE)?
((VXmlReference)c).ReferenceNode.NodeType : c.NodeType;
                            ACTION_SetTreeNodeName(tn, c);
                        c = c.Next;
                    }
            }
            tvCat.EndUpdate();
            Cursor.Current = Cursors.Default;
```

4. Search nodes using XQL expression

```
            VXmlNode n = null;
            string st = mnuSearchTextBox.Text.Trim();
            if (st.Length > 0)
            {
                decimal d = 0;
```

```
                if (CONTEXT_LIST_VIEW_ITEM == null)
                    n = CONTEXT_XML_NODE;
                else
                    n = cont.GetNode(VSLib.ConvertStringToLong(CONTEXT_LIST_VIEW_ITEM.SubItems[0].Text));

                if (n.NodeTypeCode == DEFX.NODE_TYPE_REFERENCE)
                    n = ((VXmlReference)n).ReferenceNode;

                //n = CONTEXT_XML_NODE; // cont.GetNode(VSLib.ConvertStringToLong(tvCat.SelectedNode.Name));

                VXmlNodeCollection l = null;
                try
                {
                    lbResult.Items.Clear();
                    pnQuery.Visible = true;
                    tsNodeCount.Text = "Running query...";
                    Application.DoEvents();
                    Cursor.Current = Cursors.WaitCursor;
                    long l_start = DateTime.Now.Ticks;
                    l = n.SelectNodes(st);
                    long l_end = DateTime.Now.Ticks;
                    d = l_end - l_start;
                }
                catch (VXmlException e1)
                {
                    MessageBox.Show(e1.Message, "Error", MessageBoxButtons.OK);
                }

                if (l != null)
                {
                    tsNodeCount.Text = "Selected " + l.Count.ToString() + " nodes in " + (d / 10000000).ToString("F") + "
seconds";
                    Application.DoEvents();
                    Cursor.Current = Cursors.WaitCursor;
                    lbResult.BeginUpdate();
                    for (int i = 0; i < l.Count; i++)
                    {
                        lbResult.Items.Add("<Id:" + l[i].Id.ToString("D8") + "><Name:" + l[i].Name + "><Type:" +
l[i].NodeType + ">");
                    }
                    lbResult.EndUpdate();
                }
```


mailto: ivvvsx@gmail.com

```
        else
            tsNodeCount.Text = "";
        Cursor.Current = Cursors.Default;
        pnQuery.Visible = true;
        mnuHide.Visible = true;
        SHOW_Buttons();
    }
```

VI. Contact Us

mailto:ivvvsx@gmail.com