Contents

Chapter 1

Introduction

The frames used to exchange information between the server and the client are all written in XML. They all contain the same root tag (`ClientServerXML` tag) to ensure that a frame is a part of this protocol. The root has exactly one child, the type tag, whose name gives the type of the frame. This type tag may have children or not depending on the type of the frame.

There are four frames structures :

- Some type tags need arguments. All arguments are mandatory. If an argument is missing, the frame is not valid.
- Some type tags need XML data, appended as children of the type tag. Data format depends on the type tag.
- Some type tags need arguments and data.
- Some type tags need neither arguments nor data.

Chapter 2

From client to server

2.1 Get tree

2.1.1 Description

This frame is used by the client to request the XML tree.

2.1.2 Structure

```
<ClientServerXML>
  <getTree/>
</ClientServerXML>
```

2.2 Add node

2.2.1 Description

This frame is used by the client to add a node.

2.2.2 Structure

```
<ClientServerXML>
  <addNode path="/path/to/parent" name="nodeName" type="concreteType"
     abstype="abstractType" />
</ClientServerXML>
```

2.2.3 Arguments

`path` : Path from the root to the parent node of the new node. Node names are separated by a *slash* ('/').

`name` : Name of the new node.

`type` : Concrete type of the new node.
`absType` : Abstract type of the new node.

2.2.4 Server answer on success

The server answers by sending the updated tree (see also `tree`, section 3.1, page 11) to all clients.

2.2.5 Possible errors

- The path is not valid.
- The name contains a space.
- A child node with this name already exists for this parent.
- The abstract type does not exist.
- The concrete does not exist or does not inherit from the given abstract type.

2.3 Rename node

2.3.1 Description

This frame is used by the client to rename a node. It is not considered as an error if the new name is equal to the current node name.

2.3.2 Structure

```
<ClientServerXML>
  <renameNode path="/path/to/node" name="newName" />
</ClientServerXML>
```

2.3.3 Arguments

`path` : Path from the root to the node to rename. Node names are separated by a *slash* ('/').
`name` : New name for the node.

2.3.4 Server answer on success

The server answers by sending the updated tree (see also `tree`, section 3.1, page 11) to all clients.

2.3.5 Possible errors

- The path is not valid.
- The name contains a space.
- A child node with this name already exists for this parent.

2.4 Delete node

2.4.1 Description

This frame is used by the client to delete a node.

2.4.2 Structure

```
<ClientServerXML>
  <deleteNode path="/path/to/node" />
</ClientServerXML>
```

2.4.3 Arguments

`path` : Path from the root to the node to delete. Node names are separated by a *slash* ('/').

2.4.4 Server answer on success

The server answers by sending the updated tree (see also `tree`, section 3.1, page 11) to all clients.

2.4.5 Possible errors

- The path is not valid.

2.5 Modify node

2.5.1 Description

This frame is used by the client to modify a node, which means add or modify options of a node.

2.5.2 Structure

```
<ClientServerXML>
  <modifyNode>
    <modOptions path="/path/to/node">
      (modified options)
    </modOptions>
    <addOptions path="/path/to/node">
      (new options)
    </addOptions>
  </modifyNode>
</ClientServerXML>
```

2.5.3 Arguments

`path` : Path from the root to the node to modify. Node names are separated by a *slash* ('/').

2.5.4 Data structure

Data structure is composed of two distinct sections. The first section (`modOptions`) contains all existing options to modify. The second section (`addOptions`) contains all new options to add. Both sections are facultative and `addOptions` can be set before `modOptions`. The protocol allows to have several `addOptions` and `modeOptions` sections so that it is possible to modify more than one node at a time.

2.5.5 Server answer on success

The server answers by sending the updated tree (see also `tree`, section 3.1, page 11) to all clients.

2.5.6 Possible errors

- The path is not valid.
- An option in `modOptions` section does not exist.
- An option in `addOptions` section already exists.

2.6 Get abstract types

2.6.1 Description

This frame is used by the client to request abstract types list for a specified type.

2.6.2 Structure

```
<ClientServerXML>
  <getAbstractTypes typeName="typeName" />
</ClientServerXML>
```

2.6.3 Arguments

`typeName` : Type name.

2.6.4 Server answer on success

The server answers by sending the list of the requested abstract types (see also `abstractTypes`, section 3.4, page 12).

2.6.5 Possible errors

- The given type name does not exist.

2.7 Get concrete types

2.7.1 Description

This frame is used by the client to request concrete types list for a specified abstract type.

2.7.2 Structure

```
<ClientServerXML>
  <getConcreteTypes typeName="abstractTypeName" />
</ClientServerXML>
```

2.7.3 Arguments

`typeName` : Abstract type name.

2.7.4 Server answer on success

The server answers by sending the list of the requested concrete types (see also `concreteTypes`, section 3.5, page 12).

2.7.5 Possible errors

- The given type name does not exist.

2.8 Open directory

2.8.1 Description

This frame is used by the client to open a directory on the server, read it and send its contents.

2.8.2 Structure

```
<ClientServerXML>
  <openDir path="/path/to/dir"/>
</ClientServerXML>
```

2.8.3 Arguments

`path` : Path to directory to open. This path can be absolute or relative (from default path). If the path is empty, a default path is used.

2.8.4 Server answer on success

The server answers by sending the directory contents (see section **??**, page **??**).

2.8.5 Possible errors

- The path is not valid.

2.9 Open file

2.9.1 Description

This frame is used by the client to open a case file

```
<ClientServerXML>
  <openFile filename="filename" />
</ClientServerXML>
```

2.9.2 Arguments

`filename` : Filename (and its path) to open. The path can be absolute or relative (from default path).

2.9.3 Server answer on success

The server answers by sending a ACK frame (see also `ack`, section 3.7, page 13).

2.9.4 Possible errors

- The path is not valid.

2.10 Shutdown server

2.10.1 Description

This frame is used by the client to shutdown the server

2.10.2 Structure

```
<ClientServerXML>
  <shutdownServer />
</ClientServerXML>
```

2.10.3 Server answer on success

The server never answers to this frame.

2.11 Run simulation

2.11.1 Description

This frame is used by the client to launch the simulation

2.11.2 Structure

```
<ClientServerXML>
  <runSimulation />
</ClientServerXML>
```

2.11.3 Server answer on success

The server answers by sending a ACK frame (see also `ack`, section 3.7, page 13).

2.11.4 Possible errors

- No case file open.
- A simulation is already running.

Chapter 3

From server to client

3.1 Tree

3.1.1 Description

This frame is used by the server to send the tree.

3.1.2 Structure

```
<ClientServerXML>
  <tree>
    (tree)
  </tree>
</ClientServerXML>
```

3.1.3 Data structure

Data represent the tree as returned by the simulator.

3.2 Message

3.2.1 Description

This frame is used by the server to send a message to the client.

3.2.2 Structure

```
<ClientServerXML>
  <message value="Message␣text" />
</ClientServerXML>
```

3.2.3 Arguments

`value` : Message text

3.3 Error

3.3.1 Description

This frame is used by the server to send a message describing an error to the client.

3.3.2 Structure

```
<ClientServerXML>
  <error value="Error␣message" />
</ClientServerXML>
```

3.3.3 Arguments

`value` : Error message text

3.4 Abstract types

3.4.1 Description

This frame is used by the server to send abstract types list to the client.

3.4.2 Structure

```
<ClientServerXML>
  <abstractTypes typeName="typeName" typesList="type1,␣type2,␣..." />
</ClientServerXML>
```

3.4.3 Arguments

`typeName` : Name of the type containing these abstract types.
`typesList` : Types list. Items are separated by the string ", ".

3.5 Concrete types

3.5.1 Description

This frame is used by the server to send concrete types list to the client.

3.5.2 Structure

```
<ClientServerXML>
  <concreteTypes typeName="typeName" typesList="type1,␣type2,␣..." />
</ClientServerXML>
```

3.5.3 Arguments

`typeName` : Name of the abstract type containing these concrete types.
`typesList` : Types list. Items are separated by the string ", ".

3.6 Directory contents

3.6.1 Description

This frame is used by the server to a directory contents.

3.6.2 Structure

```
<ClientServerXML>
  <dirContent path="/path/to/dir" dirs="dir1*dir2*dir3..."
      files="file1*file2*file3..." />
</ClientServerXML>
```

3.6.3 Arguments

`path` : Directory path.
`dir` : Sub-directories list. Items are separated by an asterisk.
`files` : Files list. Items are separated by an asterisk.

3.7 Acknowledgement

3.7.1 Description

This frame is used by the server to indicate that the specified command (specified by the type of its frame) succeeded.

3.7.2 Structure

```
<ClientServerXML>
  <ack type="typeToAck" />
</ClientServerXML>
```

3.7.3 Arguments

`type` : Type to acknowledge.

3.8 Non-acknowledgement

3.8.1 Description

This frame is used by the server to indicate that the specified command (specified by the type of its frame) failed.

3.8.2 Structure

```
<ClientServerXML>
  <nack type="typeToNack" />
</ClientServerXML>
```

3.8.3 Arguments

`type` : Type to non-acknowledge.

3.9 Simulation running

3.9.1 Description

This frame is used by the server to indicate that the simulation is running. Used to notify a new client that a simulation is already running.

3.9.2 Structure

```
<ClientServerXML>
  <simulationRunning />
</ClientServerXML>
```