# TelEnvyXML Parser

TELEnvyXML (TEX) reads a well formatted XML file which contains session responses to prompts received by an OpenVMS application. The TEX XML format consists of XML tags which when processed by TEX, issues commands or waits for data responses from the OpenVMS application.

For this example, the Telnet Server is an OpenVMS system running the MANMAN application.

## Consuming TEX

Usage:

Add the component to a form or service component.

Set the properties.

Configuring XML:

XML Structure

<?xml version="1.0" encoding="utf-8"?>

<Session>

<SessSeq ServerName="10.4.0.3" ServerTimeout="13000" DebugEnabled="true" LoggingEnabled="true" RecordingEnabled="true">

:: :: 🡨 TELEnvyXML Tags

</SessSeq>

</Session>

Waiting for Data:

### <ShellWaitForResponse>

Data and Attributes:

Data Regular Expression to Wait for.

Grab

Events Fired:

If defined in the calling application, the ShellWaitForResponse tag fires the OnGrabChangedEvent sending the recent buffer to the calling application as a string array.

### <WaitForData>

The following WaitForData functions are available for testing the existence of a specified string or Regex expression.

<WaitForDataRow Row=”*n*”>*Data</WaitForDataRow>*

Processes any available incoming data until a texts appear within the row that matches the specified *Pattern*, or until the specified maximum wait time period elapses (<see cref="Timeout"/>).

Data and Attributes:

Data Regular Expression to Wait for.

Row Cell row.

<WaitForDataRegion Column=”*x*” Row=”*y*” Width=”*w*” Height=”*h*” >*Data</WaitForDataRegion>*

Processes any available incoming data until a text appear within the specified screen region that matches the specified *Pattern*, or until the specified maximum wait time period elapses (<see cref="Timeout"/>).

Data and Attributes:

Data Regular Expression to Wait for.

Column Cell column.

Row Cell row.

Width Width of the region.

Height Height of the region.

<WaitForDataScreen>*Data</WaitForDataScreen>*

Processes any available incoming data until a text appear on the terminal screen that matches the specified *pattern*, or until the specified maximum wait time period elapses.

Data and Attributes:

Data Regular Expression to Wait for.

Attributes: None

The control will wait until the cursor position has changed to the screen coordinates as specified by the enclosed attributes. The control will wait until the specified ServerTimeOut is reached and will then throw a WaitCursorTimeOut exception.

### <WaitForCursor>

The control processes all data until the cursor position has changed to the screen coordinates at the specified position or until the control reaches maximum wait time as specified by ServerTimeOut. If ServerTimeOut is reached, the TeX will send a WaitCursorTimeOut exception to the application.

Requires Arguments:

Row = “X”

Column=”Y”

### Capturing Data:

The process of capturing data is actually quite simple using the TeX XML interface. Capturing data occurs when a Grab{tag} is used for scrape the Virtual Terminal control using specified coordinates for Row, Column, Size of data to be grabbed and number of lines to grab.

Data are passed back to the calling application via the OnGrabChange event.

Considerations:

Unless specified, data are returned as a string array. It is the responsibility of the calling application to perform all string parse functions on the data to consume it.

Helper classes for targeted coordinates were created to facilitate field /values cast from string to Int32 or Double.

#### TestRegion

The TeX control returns a region of text beginning at column of Width size beginning at Row for Row + Height rows. The TestRegion checks the returned text for an match of any of the data to the Regex expression (first argument of the method function). If the Regex pattern matches the region data, the function returns true, otherwise returns false.

Arguments:

Regex Expression

Column

Row

Width

Height

Returns True if the text is found in the region specified.

#### <GrabLine>

The TeX control returns the specified row of text beginning at offset “Column” of “Width” length. Data is passed to the calling application via the OnGrabChange event.

If the TeX XML attempts to parse beyond the length of the row, a TeXOutputBufferOverFlow

Exception is thrown.

Arguments:

Column :: Column starting offset

Row :: The selected row.

Width :: the return value is Column + Width length.

Assumes the current row.

#### <GrabLines>

If the TeX XML attempts to parse beyond the length of the row, a TeXOutputBufferOverFlow

Exception is thrown.

Arguments:

Column :: Column starting offset

Row :: The selected row.

Width :: the return value is Column + Width length.

Height :: The rows between the Row and Row+Height count of rows are selected.

#### <GrabInt32>

If the TeX XML attempts to parse beyond the length of the row, a TeXOutputBufferOverFlow

Exception is thrown.

If the TeX XML attempts to parse string data which cannot be cleanly cast as Int32 a TeXInvalidCast is thrown.

Arguments:

Column :: Column starting offset

Row :: The selected row.

Width :: the return value is Column + Width length.

Assumes the current row.

#### <GrabDouble>

If the TeX XML attempts to parse beyond the length of the row, a TeXOutputBufferOverFlow

Exception is thrown.

If the TeX XML attempts to parse string data which cannot be cleanly cast as Int32 a TeXInvalidCast is thrown.

Arguments:

Column :: Column starting offset

Row :: The selected row.

Width :: the return value is Column + Width length.

Assumes the current row.

#### <Expect>

Returns all data from SendCommand less the string specified by the <Expect> node.

Argument: String that is expected in the components telnet buffer.

Returns Raw string data to the calling application.

Exception: If the return value is not initialized (no match was found), the return result is a null string. Null Strings are not processed by the component and a TeXNullResultException is thrown to the application.

Implemented as:

Public GetDataBlock(string pattern){

string result = Expect(pattern);

return \_shell.ReceivedData;

}

Description of Implementation:

Return null if result doesn't contain a matched string.

It is good for the following case:

\_shell.SendCommand("ls");

string result = \_shell.Expect("prompt$ ");

In this case, the result contains command output only (and exactly the command output) (without command: "ls\n" and without prompt:"prompt$ ")

But in a more general case, the following is more probable:

SendData("directory\n")

string result = \_shell.Expect("$ ");

In this case `result` contains command: "ls\n", command output and start of the prompt: "prompt"

So returning all received data is less confusing. It contains command: "ls\n", command output and whole prompt: "prompt$ "

### Sending Data:

#### *<ShellSendCommand>*

The control sends data contained within this XML tag to the server. Data are sent without escaped control sequences.

#### <SendEnter>

The control sends data contained within this XML tag to the server. Data are sent followed by a CR. This tag control does not require data to be passed within the tag.

Data is always sent to the TeX component followed by a \n {new line} sequence. A note for MANMAN MMCURSOR users, SendEnter should be used in place of ShellSendCommand.

#### <SendTab>

The control sends data contained within this XML tag to the server. Data are sent followed by a Tab (\t). This tag control does not require data to be passed within the tag.

Data are always sent to the TeX component followed by an excaped control sequence, e.g. escape+t “\t” {tab}.

#### <SendData>

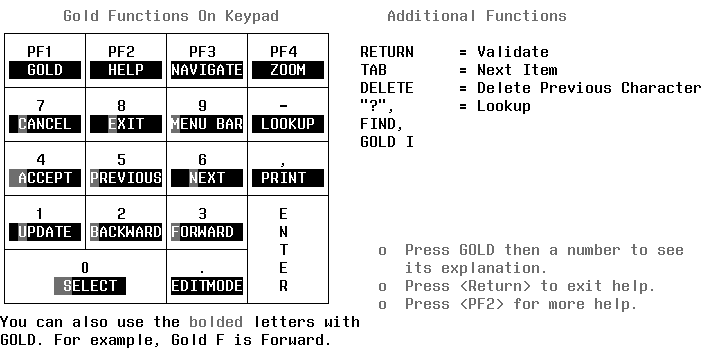
Data are transmitted to the TeX component. No special sequences or controls are sent with the data (.e.g. <CR><LF>).

TelEnvyXML Keyboard Support



Special Functions:

OpenVMS DecForms Support:



Short-Cut Navigation.

<SendPF1>



When processing this tag, >Escape + P is sent to the Server.

<SendPF2>



When processing this tag, >Escape + Q is sent to the Server.

<SendPF3>



When processing this tag, >Escape + R is sent to the Server.

<SendPF4>



When processing this tag, >Escape + S is sent to the Server.

<SendF6>

When processing this tag, >Escape + “17~” is sent to the Server.

<SendF7>

When processing this tag, >Escape + “18~” is sent to the Server.

<SendF8>

When processing this tag, >Escape + “19~” is sent to the Server.

<SendF9>

When processing this tag, >Escape + “20~” is sent to the Server.

<SendF10>

When processing this tag, >Escape + “21~” is sent to the Server.

<SendF11>

When processing this tag, >Escape + “22~” is sent to the Server.

<SendF12>

When processing this tag, >Escape + “23~” is sent to the Server.

Default Debugging Directory

Unless changed at run-time, all debug, logging and recording files are created in the c:\envy\telenvy\logs\ directory.

SessSeq Required Attributes

ServerName

ServerTimeout

DebugEnabled

LoggingEnabled

RecordingEnabled

# Programmers Reference

# ComSession Application:

Properties:

Methods:

Events:

OnNodeProcessing Before the node is processed, the node is passed to the calling application to be processed or reported.

Arguments: ComSessionNodeProcessEventArg

OnNodeProcessed After the node is processed, the node is passed to the calling application to be processed or reported.

Arguments: ComSessionNodeProcessEventArg

### <WaitForData>

The following WaitForData functions are available for testing the existence of a specified string or Regex expression.

<WaitForDataRow Row=”*n*”>*Data</WaitForDataRow>*

Processes any available incoming data until a texts appear within the row that matches the specified *Pattern*, or until the specified maximum wait time period elapses (<see cref="Timeout"/>).

Data and Attributes:

Data Regular Expression to Wait for.

Row Cell row.

When called in an application:

Returns True if the specified pattern appeared on the screen; False otherwise.

<WaitForDataRegion Column=”*x*” Row=”*y*” Width=”*w*” Height=”*h*” >*Data</WaitForDataRegion>*

Processes any available incoming data until a text appear within the specified screen region that matches the specified *Pattern*, or until the specified maximum wait time period elapses (<see cref="Timeout"/>).

Data and Attributes:

Data Regular Expression to Wait for.

Column Cell column.

Row Cell row.

Width Width of the region.

Height Height of the region.

When called in an application:

Returns True if the specified pattern appeared on the screen; False otherwise.

<WaitForDataScreen>*Data</WaitForDataScreen>*

Processes any available incoming data until a text appear on the terminal screen that matches the specified *pattern*, or until the specified maximum wait time period elapses.

Data and Attributes:

Data Regular Expression to Wait for.

Attributes: None

When called in an application:

Returns True if the specified pattern appeared on the screen; False otherwise.

The control will wait until the cursor position has changed to the screen coordinates as specified by the enclosed attributes. The control will wait until the specified ServerTimeOut is reached and will then throw a WaitCursorTimeOut exception.

Appendix:

## Special Function Keys

**Key Numeric ANSI Mode Numeric VT52 Mode**

**Application Application**

0 0 Ss3p 0 Esc?p

1 1 Ss3q 1 Esc?q

2 2 Ss3r 2 Esc?r

3 3 Ss3s 3 Esc?s

4 4 Ss3t 4 Esc?t

5 5 Ss3u 5 Esc?u

6 6 Ss3v 6 Esc?v

7 7 Ss3w 7 Esc?w

8 8 Ss3x 8 Esc?x

9 9 Ss3y 9 Esc?y

- (minus) Ss3m - Esc?m

, (comma) Ss3l , Esc?l

. (period) Ss3n . Esc?n

Enter Cr or Ss3M Cr or Esc?M

CrLf CrLf

PF1 Ss3P Ss3P EscP EscP

PF2 Ss3Q Ss3Q EscQ EscQ

PF3 Ss3R Ss3R EscR EscR

PF4 Ss3S Ss3S EscS EscS

## Exceptions:

### TeXWaitCursorTimeOut A wait condition was sent to the TeX component expecting that the cursor will be positioned at specific screen coordinates. The cursor control data returned from the server did not match the expected coordinates.

### TeXWaitDataRegionTimeOut A wait condition was sent to the TeX component for a data element or a Regex expression and the data returned from the server did not match for the expected data.

### TeXNullResultException A non-null string was expected but what was returned was a null value. This is probably caused by a mis-match of an Expect tag.

### TeXXMLInvalidTag A tag was found in the TeX XML file that is not supported.

### TeXConditionOutOfRange A Condition tag was found which is not supported.

Supported Condition Types:

case IfConditionType.EmptyIfCondition:

case IfConditionType.CursorAtPosition:

case IfConditionType.CursorWithinRegion:

case IfConditionType.DataAtCursorPosition:

case IfConditionType.DataBeforeCursorPosition:

case IfConditionType.DataWithinRegion:

# Application Configuration

The configuration properties can be customized. The application configuration properties are located in the “<directory>”.

UseEventLog When a file is processed, the file name and log information will be recorded in an application event log.

InputDirectory XML files are processed from this directory. TELEnvy can be configured to proactively monitor the directory and process new files or manually execute files from this directory.

OutputDirectory TELEnvy XML processed files are renamed to this directory.

ArchiveDirectory The directory where completed XML files are written.

RecorderDirectory The directory where TELEnvy’s recorder files are stored.

LogDirectory The directory where TELEnvy’s run-time log files are stored.

LogLevel The logging level that will be used when processing TELEnvy XML Files.

Log Type Level Description

* Verbose [0] Extensive Logging
* Debug [1] Log all message useful for debugging.
* Info [2] Log Informative messages.
* Error [4] Log only Errors
* Off [8] Turn-off logging.

The <SessSeq> node is a required node to process the XML responses. Please refer to SessSeq Required Attributes

section of this guide.

## Example Configuration:

### MANMAN MAC419

<?xml version="1.0" encoding="utf-8"?>

<Session>

<SessSeq ServerName="MMTest" ServerTimeout="13000" DebugEnabled="true" LoggingEnabled="true" RecordingEnabled="true">

<Expect>([Uu]sername)|([Ll]ogin): ?</Expect>

<SendEnter>peer</SendEnter>

<Expect>[Pp]assword: ?</Expect>

<SendEnter>timothye</SendEnter>

<Expect>MMtest> </Expect>

<SendEnter>manman</SendEnter>

<WaitForData>PLANT CODE</WaitForData>

<SendEnter>supplies</SendEnter>

<WaitForData>COMMAND.\*\\*</WaitForData>

<SendEnter>om c 419</SendEnter>

<WaitForData>SHIP-TO CUSTOMER NUMBER</WaitForData>

<SendEnter>049999</SendEnter>

<WaitForData>NEW SHIP-TO CUSTOMER NAME?</WaitForData>

<SendEnter>Test Change Shipto Name</SendEnter>

<WaitForData>SHIP-TO CUSTOMER NUMBER</WaitForData>

<SendEnter>E</SendEnter>

<!--End Command -->

<WaitForData>PLANT CODE</WaitForData>

<SendEnter>E</SendEnter>

<!-- Exit MANMAN -->

<WaitForData>EXITING MANMAN </WaitForData>

<Expect>MMtest></Expect>

<SendEnter>LOGOUT</SendEnter>

<!--Log Out -->

</SessSeq>

</Session>