Text Terminal Support via XLM Scripting

PageManager Pro XML Scripting Format

The format of the protocol script is XML-based. XML is a formatted ASCII text that makes it possible to define special data structures.

The scripting file describes a certain protocol and the steps necessary for sending a message.
The protocol description file consists of a number of items where each item represents one operation, e.g. the sending or receiving of certain data.

A sample file describing a very simple text-bases messaging protocol might look like follows.

```
<?xml version="1.0"?>

<mm_protocol name="Scripting - Sample" version="1.0">

<item
        name="start"
        operation="receive">
                <received goto="enter_number">
Welcome to My Service. Enter number ...
                </received>
                <received_else goto="end" result="-1"/>
                <received_nothing goto="end" result="-1"/>
</item>

<item
        name="enter_number"
        operation="send">
                <send goto="text_prompt">
[[MM_MSG_TO]][[MM_MSG_ASCII(0D)]]
                </send>
</item>

<item
        name="text_prompt"
        operation="receive">
                <received goto="enter_text">
Number correct. Enter message ...
                </received>
                <received_else goto="end" result="-1"/>
                <received_nothing goto="end" result="-1"/>
</item>

<item
        name="enter_text"
        operation="send">
                <send goto="again">
[[MM_MSG_MESSAGEBODY]][[MM_MSG_ASCII(0D)]]
                </send>
</item>

<item
        name="again"
        operation="receive">
                <received goto="not_again">
Message accepted. Do you want to send another message (Y or N)?
```

```
                </received>
                <received_else goto="end" result="-1"/>
                <received_nothing goto="end" result="-1"/>
</item>

<item
        name="not_again"
        operation="send">
                <send goto="end" result="0">
N[[MM_MSG_ASCII(0D)]]
                </send>
</item>

<item
        name="end"
        operation="terminate">
</item>

</mm_protocol>
```

The above file describes a protocol that will prompt the client for a recipient number and for the message text in order to send the message. The information flow might look like follows.

Dialing.
Connecting.
Receiving: *Welcome to My Service. Enter number ...*
Sending: *1234567*<CR>
Receiving: *Number correct. Enter message ...*
Sending: *Hello world.*<CR>
Receiving: *Message accepted. Do you want to send another message (Y or N)?*
Sending: *N*<CR>
Hang up call.

Structure of the Protocol Definition File

In the following table all parts of the protocol description file are described.

mm_protocol	This element specifies the wrapper of the protocol description. It contains all the protocol items. **Attributes:** name: The name of the protocol. version: The version number of this protocol description. At the moment only 1.0 is available.
item	This element is a sub-element of mm_protocol and represents a protocol entity, e.g. the sending or receiving of a data packet. **Attributes:**

	name: The mane of the item. This name is used in order to address the item from out of other items. The name *start* is reserved for representing the start item and the name *end* is reserved for representing the end item. operation: Specifies the operation this item is used for. The following operations are supported: *send*: This item is used to send a data packet to the server. *receive*: This item is used to receive a data packet from the server. next: This items forces the messaging process to continue with the next message if available. terminate: This item is used to terminate the protocol session. The component will close the connection afterwards.
received	This element is a sub-element of the item element if the operation attribute is *receive*. The content of this element represents a text according to the data expected to receive here. A receive item can contain more than one received elements. These elements work like a switch. Depending on the actual data received the one or the other action will be triggered. **Attributes:** goto: The goto attribute specifies the item name of the next item that should be triggered in case the according data was received. result: This attribute specifies the result code that should be returned if the session would be terminated in the next step. This result code can be overwritten by sub-sequential result attributes. The result must be 0 if no error has occurred.
received_else	This element is a sub-element of the item element if the operation attribute is *received*. This element has basically the same functionality as the received element but it will be triggered if a text was received but no other received element does contain fitting data.

	This element has no content. **Attributes:** This element has the same attributes as the received element.
received_nothing	This element is a sub-element of the item element if the operation attribute is *received*. This element has basically the same functionality as the received element but it will be triggered if no data was received during the wait-time period. This element has no content. **Attributes:** This element has the same attributes as the received element.
send	This element is a sub-element of the item element if the operation attribute is *send*. The content of this element represents the text data to be sent to the server. The following keywords are possible in the text data (the keywords will be translated into the appropriate message part during the communication process): *[[MM_MSG_TO]]*: The recipient number. *[[MM_MSG_FROM]]*: The sender number or sender id. *[[MM_MSG_MESSAGEBODY]]*: The message text. *[[MM_MSG_ASCII(DD)]]*: An ASCII character where DD means the hexadecimal encoded character, e.g. [[MM_MSG_ASCII(33)]] will be translated to the ASCII character '3'. Further fields, e.g. user name, password, etc., can be hard-coded in the protocol description file as plain text. One item can contain only one send element. **Attribute:** goto: The goto attribute specifies the item name of the next item that should be triggered in case the according data

	was received. result: This attribute specifies the result code that should be returned if the session would be terminated in the next step. This result code can be overwritten by sub-sequential result attributes. The result must be 0 if no error has occurred.
next	This element is a sub-element of the item element if the operation attribute is *next*. The attributes of this element apply if a next message was found. **Attribute:** goto: The goto attribute specifies the item name of the next item that should be triggered in case the according data was received. result: This attribute specifies the result code that should be returned if the session would be terminated in the next step. This result code can be overwritten by sub-sequential result attributes. The result must be 0 if no error has occurred.
nonext	This element is a sub-element of the item element if the operation attribute is *next*. The attributes of this element apply if no next message could be found, i.e. if all messages have been processed and no message is left anymore. **Attribute:** goto: The goto attribute specifies the item name of the next item that should be triggered in case the according data was received. result: This attribute specifies the result code that should be returned if the session would be terminated in the next step. This result code can be overwritten by sub-sequential result attributes. The result must be 0 if no error has occurred.

First steps

Each XML script must be located in the same directory as PageManager Pro. The name specified within the XML scripting file will become the name of the base service later.

In order to test XML scripting files just while creating them the according files can be opened in the Internet Explorer version 6.0. This browser will show the XML file in a tree structure or it will provide error information if the XML file is malformed.

Orange Terminal Example for England
(Terminal Phone: 44 07973 100602)

```
<?xml version="1.0"?>

<mm_protocol name="Scripting - Orange UK" version="1.0">

<item name="start"   operation="receive">
        <received goto="choose_option">
:- Exit
        </received>
        <received_else goto="end" result="-1"/>
        <received_nothing goto="end" result="-1"/>
</item>

<item name="choose_option"  operation="send">
        <send goto="pager_number_prompt">
s
        </send>
</item>


<item name="pager_number_prompt" operation="receive">
        <received goto="enter_number">
Enter destination Orange or Hutchison Pager number and press return
        </received>
        <received_else goto="end" result="-1"/>
        <received_nothing goto="end" result="-1"/>
</item>

<item name="enter_number"         operation="send">
        <send goto="text_prompt">
[[MM_MSG_TO]][[MM_MSG_ASCII(0D)]]
        </send>
</item>

<item name="text_prompt" operation="receive">
```

```
        <received goto="enter_text">
Type your message (160 characters max) and press return to send
        </received>
        <received_else goto="end" result="-1"/>
        <received_nothing goto="end" result="-1"/>
</item>

<item name="enter_text" operation="send">
        <send goto="again">
[[MM_MSG_MESSAGEBODY]][[MM_MSG_ASCII(0D)]]
        </send>
</item>

<item name="again" operation="receive">
        <received goto="next_message" result="0">
Message accepted - thankyou
        </received>
        <received_else goto="end" result="-1"/>
        <received_nothing goto="end" result="-1"/>
</item>

<item
        name="next_message" operation="next">
                <next goto="choose_option" result="0"/>
                <nonext goto="not_again" result="0"/>
</item>

<item name="not_again" operation="send">
        <send goto="end" result="0">
E
        </send>
</item>

<item name="end" operation="terminate">
</item>

</mm_protocol>
```