XML Messaging API
Technical Documentation

Site version: 6.8.1+
Document version: 2.8
Author: Greg J Preece
Author: Kris Bloe
txttoolstechies@blackboard.com
Date: 02/05/2012

Contents

XML Connector for Messaging

What is the XML Connector?

The XML Connector allows web and application developers a fast, flexible and standards-based method for sending and receiving SMS messages as well as status updates for sent messages. XML is delivered to the server using the standard HTTP 1.1/1.0 POST to a web form. All responses from the connector are also provided to the requesting client as XML.

The required structure of the XML, as well as that of the expected response from the connector, is described in the following section.

For more information on XML see the W3C XML site at:

http://www.w3.org/XML/

For more information on HTTP 1.1 see the relevant RFC

RFC2616: Hypertext Transfer Protocol -- HTTP/1.1
http://www.ietf.org/rfc/rfc2616.txt

NB: If you are intending to use the newer HTTP 1.1 rather than HTTP 1.0, please ensure that your client or code is capable of handling the Transfer-Encoding:Chunked header, as large responses will be chunked by the server.

Requirements for Using the XML Messaging Connector

In order to send or receive messages through the XML messaging connector, you first build the required XML document, URI-encode it to ensure there are no problems during transmission, and then send the document to the XML connector over HTTP 1.1/1.0.

The complete XML structure for request **MUST** be contained in a <Request> tag and **MUST** contain an Authentication structure and one or more sub-elements. Please correctly escape any character fields using <![CDATA[]]> tags, and URI-encode your transmissions, to prevent parse errors. Please also note that **XML is case sensitive** and tags should be used as shown.

NB: ConnectTxt accounts must be enabled for use with the XML connector. Contact txttoolssupport@blackboard.com if you need to have your account enabled.

It is recommended to use the SSL 128-bit encrypted version of the XML connector. The SSL connector version is available at:
https://www.txttools.co.uk/connectors/XML/xml.jsp

If your client cannot handle SSL, the unencrypted address for the XML messaging connector is:

http://www.txttools.co.uk/connectors/XML/xml.jsp

The XML payload should be sent via POST to the appropriate connector URL, as the value of a POST variable called XMLPost.

Please note: When connecting to the XML messaging connector, we ask that you please set your User-Agent string to be the name of your integration product or institution, or your ConnectTxt username if this is not possible. This enables us to identify which users are having problems connecting, and assist them if necessary.

The XML Messaging Connector Test Page

If you wish to test your constructed XML manually before using it as part of an automatic system, you can use the XML connector test page to do so. Open your chosen browser and go to:

https://www.txttools.co.uk/connectors/XML/xml.jsp?test=true

This will display a very basic page containing only a textbox and a submit button. If you cut and paste your XML into this box, and click the submit button, it will be run on the live site, and the result displayed onscreen.

(Mozilla Firefox users may need to click CTRL+U to open the returned XML.)

Please note: This test page operates on the live site. If you put an XML request into it asking that a message be sent, that message **will** be sent.

The <Request> Element

The <Request> element is the root tag of the XML document. All XML sent to the connector must be contained within a <Request> tag, or you will receive an error message in response.

The <Authentication> Element

Each XML structure sent to the server **MUST** contain the authentication structure. The authentication structure, contains two required elements <Username> and <Password>. These contain the username and password of the ConnectTxt account you want to send/receive messages through.

Please note: ConnectTxt accounts must be enabled for use with the XML connector. Contact txttoolssupport@blackboard.com if you need to have your account enabled.

Authentication Structure Elements:

Element	Description	Required
Authentication	Container element.	**YES**
Username	The Username for authentication, it should contain a text string of the username provided.	**YES**
Password	The Password for authentication, it should contain the password provided.	**YES**

Example Authentication Structure:

```
<Authentication>
      <Username><![CDATA[MyUsername]]></Username>
      <Password><![CDATA[MyPassword]]></Password>
</Authentication>
```

Sending an SMS Message

To send a message, you need to include one or more <Message> elements. This element should include the items necessary for message delivery. Required fields are <MessageText>, <Type> and at least one <Phone> or <GroupId> tag.

<Message> Structure Elements:

Element	Description	Required
Message	The top level element for this structure.	**YES**
MessageText	The text of the message to send.	**YES**
Phone	The phone number including the international dialing prefix of each recipient should be contained in this tag. (e.g. +447777777777 or 00447777777777) Please Note the '+' character in a phone number can cause encoding problems and should be replaced with %2b if you are experiencing delivery problems. Alternatively the 0044... or 07... forms may be used. Please remember to use <![CDATA[]]> tags to encapsulate character data.	**YES (if no GroupId tags present)**
GroupId	The id of the group in your ConnectTxt addressbook that you wish to send a message to. (This is the numerical ID of the group, not the group's textual name.)	**YES (if no Phone tags present)**
Type	The type of message, 1 indicates bulk, and 2 indicates reverse charged. In most cases, you will want to use 1.	**YES**
ScheduleTimeUTCSecs	A UTC timestamp (the time in seconds since 00:00:00 Jan 1 1970) showing when the message should be delivered.	NO
UniqueID	A Unique ID to assign to this message. This is not used by our system, but is passed back in the connector's response and can be used in delivery reporting information.	NO
From	If you have the appropriate permission enabled on your account (contact support to find out), this field allows you to change the message source shown on the recipient's mobile phone. If this field can contain either a phone number or up to 11 alphanumeric characters.	NO
SuppressUnicode	If this tag is included and set to true, it forces the connector to send the message in the GSM 03.38 alphabet. Any characters outside this alphabet that would normally result in a unicode/ UTF-8 message being sent will be suppressed – i.e. they will not display correctly on the receiving phone.	NO

ReportingLevel	Tells the server what level of detail you would like in the response. The default value is "FULL", and other possible values are "TICKETS" and "MINIMAL". A "FULL" response will include all available data, including status codes, unique IDs, message text and phone numbers. A "TICKETS" response will include only ticket numbers for each message, and phone numbers if they are available. A "MINIMAL" response will include only a count of the number of recipients for the message, and whether or not the send was successful. This could be useful when sending to a GroupId containing thousands of contacts. Examples of all three response types are below.	NO

Example Message Request:

```
<Message>
      <MessageText><![CDATA[The Message Text]]></MessageText>
      <Phone><![CDATA[+441234567890]]></Phone>
      <GroupId>12345</GroupId>
      <Type>1</Type>
      <ScheduleTimeUTCSecs>1234567890</ScheduleTimeUTCSecs>
      <UniqueID><![CDATA[Just an ID]]></UniqueID>
      <From><![CDATA[GregJPreece]]></From>
      <SuppressUnicode>FALSE</SuppressUnicode>
      <ReportingLevel><![CDATA[FULL]]></ReportingLevel>
</Message>
```

The resulting <MessageStatus> structure will contain the message ticket and initial status. It will also include the details of the message itself, so the receiving system can easily identify source messages and associate them with the tickets provided for retrieving status information.

Full-response reporting:

This first example shows the response you will receive if <ReportingLevel> is set to "FULL":

Element	Description	Always Included
MessageStatus	Top level element for this structure.	**YES**
MessageText	The text of the SMS message that was sent.	**YES**
Ticket	The message ticket that was provided when a message was sent.	**YES**
Status	The numeric status code for this message.	**YES**
StatusMessage	The textual status information for this message. *N.B this text is subject to change.*	**YES**

Phone	The phone number the message was sent to.	**YES**
UniqueID	If you designated a UniqueID for the message in your request, it will be passed back in this element.	Only if specified in request

Example response with full reporting:

```
<MessageStatus>
      <MessageText><![CDATA[The Message Text]]></MessageText>
      <Phone><![CDATA[+441234567890]]></Phone>
      <UniqueID><![CDATA[Just an ID]]></UniqueID>
      <Ticket>123</Ticket>
      <Status>0</Status>
      <StatusMessage>
            <![CDATA[Queued For Delivery]]>
      </StatusMessage>
</MessageStatus>
```

(You will receive one of these blocks for each individual SMS sent)

Below is the response of the same message again, but with ReportingLevel set to “TICKETS”:

Tickets-only response structure:

Element	Description	Always Included
MessageStatus	Top level element for this structure.	**YES**
Ticket	The message ticket that was provided when a message was sent.	**YES**
Phone	The phone number the message was sent to.	**YES**

Example response with reporting set to tickets-only:

```
<MessageStatus>
      <Phone><![CDATA[+441234567890]]></Phone>
      <Ticket>123</Ticket>
</MessageStatus>
```

(You will receive one of these blocks for each individual SMS sent.)

Finally, here is the response for that message one more time, but with ReportingLevel set to “MINIMAL”:

Minimal ticket response structure:

Element	Description	Always Included
MessageStatusSummary	The top level element for this structure.	**YES**
RecipientsSize	The total number of SMS messages sent.	**YES**
Success	True if the messages were accepted by ConnectTxt, otherwise false.	**YES**

Example Response with minimal reporting:

```
<Response>
      <MessageStatusSummary>
            <RecipientsSize>17397</RecipientsSize>
            <Success>true</Success>
      </MessageStatusSummary>
</Response>
```

(You will only ever receive one MessageStatusSummary block for all messages sent in a request)

Getting Status Updates

To fetch status updates for messages previously sent through the XML connector, you need to send a <RequestStatus> element in your XML request. The only valid sub-element is <Ticket> which should contain the message ticket assigned when a message was sent. Each <Ticket> element requires its own <RequestStatus> parent element.

As with sending messages, the response will contain a numeric status code (See Status Response Codes table below) and a textual status message.

PLEASE NOTE: The textual status message may change without notice. Status updates should use the numeric status code rather than the status message to identify delivery status.

Status Request Structure:

Element	Description	Required
RequestStatus	Top level element for this structure.	**YES**
Ticket	The message ticket that was provided when the message was sent.	**YES**

Example Status Request:

```
<RequestStatus>
      <Ticket>123</Ticket>
</RequestStatus>
```

Response Structure:

Element	Description	Always Included
MessageStatus	Top level element for this structure.	**YES**
Ticket	The message ticket that was provided when a message was sent.	**YES**
Status	The numeric status code for this message.	**YES**
StatusMessage	The textual status information for this message. *N.B this text is subject to change.*	**YES**
Phone	The phone number the message was sent to.	**YES**

Example Response:

```
<MessageStatus>
      <Ticket>123</Ticket>
      <Status>5</Status>
      <StatusMessage>
            <![CDATA[Delivered To Handset]]>
      </StatusMessage>
      <Phone><![CDATA[+441234567890]]></Phone>
</MessageStatus>
```

Status Codes:

Status Code	Meaning
-1	Message failed at ConnectTxt due to insufficient message credits.
0	Message is in the queue at ConnectTxt, waiting to be sent (usually immediately.)
1	Message has been delivered to the aggregator upstream and is on its way.

2	A fatal error occurred during delivery. The message cannot be delivered.
3	Message received by phone network.
4	Delivery failed. The message has not been delivered, but the network will continue to retry until the message expires.
5	Message received by handset.
6+	A fatal error occurred during delivery. The message cannot be delivered.

Retrieving Inbound Messages

The RetrieveInbound structure is used to request any messages on the server that have been queued for delivery to your account.

Retrieve Inbound Request Structure:

Element	Description	Required
RetrieveInbound	Top level element for this structure.	**YES**
RetrieveSinceUTCMillis	A UTC timestamp (milliseconds since Jan 1 1970) that messages should be retrieved after. By default, no timestamp is set, and the latest messages are retrieved instead. If your system cannot handle millisecond timestamps, use RetrieveSinceUTCSecs instead.	**OPTIONAL**
RetrieveSinceUTCSecs	A UTC timestamp (seconds since Jan 1 1970) that messages should be retrieved after. By default, no timestamp is set, and the latest messages are retrieved instead. RetrieveSinceUTCMillis is the recommended tag, but if your system cannot support millisecond timestamps, use this tag instead.	**OPTIONAL**
RetrieveType	Two possible values - 'ALL' gets all messages, whereas 'UNREAD' only fetches messages which have not already been read. ('UNREAD' was the only behaviour option in the previous version of the connector, and is the current default option as a result.)	**OPTIONAL**
RetrieveNumber	The number of messages to fetch from the system. (Maximum 50, default 50.)	**OPTIONAL**

Please Note: Unread messages are marked as read once they have been fetched through the XML connector.

Example Retrieve Inbound Request:

```
<RetrieveInbound>
      <RetrieveSinceUTCMillis>1234567890123</RetrieveSinceUTCMillis>
      <RetrieveType><![CDATA[UNREAD]]></RetrieveType>
      <RetrieveNumber>10</RetrieveNumber>
</RetrieveInbound>
```

While the <RetrieveType> and <RetrieveNumber> are fairly self explanatory, it is worth taking a moment to fully explain the effects of <RetrieveSinceUTCMillis> or <RetrieveSinceUTCSecs>. Normally, if you specify only <RetrieveType> and/or <RetrieveNumber>, the system will search backwards in time from the present for messages – i.e. you will get the most recently received messages.

However, if you specify a timestamp in either <RetrieveSinceUTCMillis> or <RetrieveSinceUTCSecs>, the system will search for messages from that timestamp **forwards**. So if you used the example above, the system would return the first 10 messages that arrived after the 1234567890123 timestamp.

This is done so that users accessing the API via an automated system can make a message request, store the timestamp of the last message received (or the time at which they make the request), and then supply that timestamp as a parameter the next time they request messages – 'get everything since my last check.'

InboundMessage Response Structure:

Element	Description	Required
InboundMessage	Top level element for this structure.	**YES**
Ticket	The internal ticket for this message.	**YES**
MessageText	The text of the message.	**YES**
Phone	The phone number of the sender.	**YES**
ReceivedTimeUTCMillis	A UTC timestamp in milliseconds showing when the message was received at ConnectTxt. You will receive this tag if you used <RetrieveSinceUTCMillis> in your message request.	**YES (See description.)**
ReceivedTimeUTCSecs	A UTC timestamp in seconds showing when the message was received at ConnectTxt. You will receive this tag if you used <RetrieveSinceUTCSecs> in your message request.	**YES (See description.)**
Destination	The phone number that the message was sent to.	**YES**

Example Response:

```
<InboundMessage>
      <Ticket>1278</Ticket>
      <MessageText><![CDATA[Example]]></MessageText>
      <Phone><![CDATA[+4471234567890]]></Phone>
      <ReceivedTimeUTCMillis>1234567890123</ReceivedTimeUTCMillis>
      <Destination><![CDATA[88020]]></Destination>
</InboundMessage>
```

If the system detects that there are more messages in the set to be retrieved, then it will also return a <MessagesLeftInSet> element. For example, if a request were made for all the latest unread messages, and 60 unread messages were present in the inbox, the ConnectTxt system would return the first 50 messages, then return a <MessagesLeftInSet> element with the value 10.

This, again, is designed to prevent messages in the inbox from being missed due to the returned data set being too large.

MessagesLeftInSet Response Structure:

Element	Description	Always Included
MessagesLeftInSet	Single-element structure, holds the number of messages still to be returned from the XML connector.	**NO**

Example Response with MessagesLeftInSet element:

```
<InboundMessage>
      <Ticket>1278</Ticket>
      <MessageText><![CDATA[Example]]></MessageText>
      <Phone><![CDATA[+447777777776]]></Phone>
      <ReceivedTimeUTCMillis>1234567890123</ReceivedTimeUTCMillis>
      <Destination><![CDATA[88020]]></Destination>
</InboundMessage>

<MessagesLeftInSet>1</MessagesLeftInSet>
```

Getting Account Details

When using ConnectTxt accounts via the XML connector, you may wish to know how how many message credits you have remaining on a given account, or how many messages have been sent through it. These basic account statistics are available for querying via the <AccountDetails> element. This structure is rather new, so it is worth noting that previous structures will continue to work.

Account Details Request Structure:

Element	Description	Required
AccountDetails	Top level element for this structure.	**YES**
GetAccountDetail s	Gets the number of message credits remaining, used on this account and the account type. Boolean field - valid values are TRUE/FALSE (though naturally you'll only ever use TRUE).	**YES**

Example Detail Request:

```
<AccountDetails>
      <GetAccountDetails>
            <![CDATA[TRUE]]>
      </GetAccountDetails>
</AccountDetails>
```

Account Details Response Structure:

Element	Description	Always Included
AccountDetail	Top level element for this structure.	**YES**
MessagesRemainin g	Holds the number of message credits remaining on this account.	**YES**
MessagesUsed	Holds the number of message credits used on this account.	**YES**
AccountType	0 - Invoiced (Unrestricted.) 1 - Prepay (MessagesRemaining cannot go below 0.)	**YES**

Example Detail Response:

```
<AccountDetail>
      <MessagesRemaining>50</MessagesRemaining>
      <MessagesUsed>15</MessagesUsed>
      <AccountType>1</AccountType>
</AccountDetail>
```

Inbox Counts

You can now request a count of the number of read/unread SMS messages in your ConnectTxt inbox. This is useful for applications that wish to read the inbox without necessarily caching all of it locally.

Inbox Count Request Structure:

Element	Description	Always Included
GetInboxCounts	This is a self-closing tag representing the request for count info, and as such is the only tag required.	**YES**

Example Count Request:

```
<GetInboxCounts />
```

Inbox Count Response Structure:

Element	Description	Always Included
InboxCounts	Top level tag for this structure	**YES**
NumberNew	The number of new (unread) messages in your ConnectTxt inbox.	**YES**
NumberRead	The number of read messages in your ConnectTxt inbox.	**YES**
TotalMessages	Convenience field showing total number of messages in your ConnectTxt inbox.	**YES**

```
<Response>
     <InboxCounts>
          <NumberNew>12</NumberNew>
          <NumberRead>38</NumberRead>
          <TotalMessages>50</TotalMessages>
     </InboxCounts>
</Response>
```

Example Request and Response

Below is a full example request to the connector where two messages are sent, three inbound are retrieved, a status update is requested for one message that is outstanding, and account details are retrieved.

PLEASE NOTE: When copy-pasting the text below, you may need to replace the double quotes as matching quotes may not be understood by the XML.

Example Request:

```
<?xml version='1.0'?>
<Request>
      <Authentication>
            <Username><![CDATA[MyUsername]]></Username>
            <Password><![CDATA[MyPassword]]></Password>
      </Authentication>

      <Message>
            <MessageText>
                  <![CDATA[This is a test message.]]>
            </MessageText>
            <Phone><![CDATA[+447777777777]]></Phone>
            <Type>1</Type>
            <ScheduleTimeUTCSecs>1234567890</ScheduleTimeUTCSecs>
            <UniqueID><![CDATA[Techies]]></UniqueID>
            <ReportingLevel><![CDATA[FULL]]></ReportingLevel>
      </Message>

      <Message>
            <MessageText>
                  <![CDATA[This is a another test message.]]>
            </MessageText>
            <Phone><![CDATA[+447777777777]]></Phone>
            <Phone><![CDATA[+447777777778]]></Phone>
            <Type>1</Type>
      </Message>

      <RetrieveInbound>
            <RetrieveSinceUTCMillis>1234567890123</RetrieveSinceUTCMillis>
            <RetrieveType><![CDATA[UNREAD]]></RetrieveType>
            <RetrieveNumber>3</RetrieveNumber>
      </RetrieveInbound>

      <RequestStatus>
            <Ticket>1234</Ticket>
      </RequestStatus>

      <AccountDetails>
            <GetAccountDetails>
                  <![CDATA[TRUE]]>
            </GetAccountDetails>
      </AccountDetails>

      <GetInboxCounts />
</Request>
```

Example Response:

```
<?xml version='1.0'?>
<Response>
      <MessageStatus>
            <MessageText>
                  <![CDATA[This is a test message.]]>
            </MessageText>
            <Phone><![CDATA[+447777777777]]></Phone>
            <Type>1</Type>
            <Ticket>1235</Ticket>
            <Status>0</Status>
            <StatusMessage>
                  <![CDATA[Queued For Delivery]]>
            </StatusMessage>
      </MessageStatus>

      <MessageStatus>
            <MessageText>
                  <![CDATA[This is a another test message.]]>
            </MessageText>
            <Phone><![CDATA[+447777777777]]></Phone>
            <Type>1</Type>
            <Ticket>3236</Ticket>
            <Status>0</Status>
            <StatusMessage>
                  <![CDATA[Queued For Delivery]]>
            </StatusMessage>
      </MessageStatus>

      <MessageStatus>
            <MessageText>
                  <![CDATA[This is a another test message.]]>
            </MessageText>
            <Phone><![CDATA[+447777777778]]></Phone>
            <Type>1</Type>
            <Ticket>3237</Ticket>
            <Status>0</Status>
            <StatusMessage>
                  <![CDATA[Queued For Delivery]]>
            </StatusMessage>
      </MessageStatus>

      <InboundMessage>
            <Ticket>1278</Ticket>
            <MessageText>
                  <![CDATA[But can I send to Myself?]]>
            </MessageText>
            <Phone><![CDATA[+447777777777]]></Phone>
            <ReceivedTimeUTCMillis>123456123456</ReceivedTimeUTCMillis>
            <Destination><![CDATA[88020]]></Destination>
      </InboundMessage>

      <InboundMessage>
            <Ticket>3122</Ticket>
            <MessageText><![CDATA[Message 2]]></MessageText>
            <Phone><![CDATA[+447777777777]]></Phone>
            <ReceivedTimeUTCMillis>123456123456</ReceivedTimeUTCMillis>
            <Destination><![CDATA[88020]]></Destination>
      </InboundMessage>

      <InboundMessage>
            <Ticket>3343</Ticket>
```

```
          <MessageText><![CDATA[Message 3]]></MessageText>
          <Phone><![CDATA[+447777777777]]></Phone>
          <ReceivedTimeUTCMillis>123456123456</ReceivedTimeUTCMillis>
          <Destination><![CDATA[88020]]></Destination>
     </InboundMessage>

     <MessagesLeftInSet>5</MessagesLeftInSet>

     <MessageStatus>
          <Ticket>1234</Ticket>
          <Status>5</Status>
          <StatusMessage>
               <![CDATA[Delivered To Handset]]>
          </StatusMessage>
     </MessageStatus>

     <AccountDetail>
          <MessagesRemaining>50</MessagesRemaining>
          <MessagesUsed>15</MessagesUsed>
          <AccountType>0</AccountType>
     </AccountDetail>

     <InboxCounts>
          <NumberNew>12</NumberNew>
          <NumberRead>38</NumberRead>
          <TotalMessages>50</TotalMessages>
     </InboxCounts>
</Response>
```

Error Codes

The following error codes may be seen when connecting to ConnectTxt.

Error code	**Description**
600	You did not specify the XMLPost parameter, or did not populate it with XML.
601	Invalid credentials.
602	One of the XML elements is null or blank.
603	I/O error - please contact txttoolssupport@blackboard.com for advice.
604	XML parse error - Check your XML tags are correct and ensure you are using CDATA containers.
605	Used in XML addressbook API, not relevant.
606	Internal server error. Contact ConnectTxt for support.
607	Invalid message type, or you are attempting to override your message type and do not have permission to do so. Please check your XML and try again.
608	Invalid binary header in transmission.

Error response structure:

Element	**Description**	**Always Included**
Error	Top level element for this structure.	**YES**
ErrorMessage	The error message.	**YES**
ErrorCode	The error code (e.g. 604.)	**YES**

Example Request (601):

```
<?xml version='1.0'?>
<Request>
      <Authentication>
            <Username><![CDATA[MyUsername]]></Username>
            <Password><![CDATA[DodgyPassword]]></Password>
      </Authentication>

      <AccountDetails>
            <GetAccountDetails>
                  <![CDATA[TRUE]]>
            </GetAccountDetails>
      </AccountDetails>
</Request>
```

Example Response (601):

```
<?xml version='1.0'?>
<Response>
      <Error>
            <ErrorMessage>Invalid login details, check your details are correct
      and that you have permission to use the XML API</ErrorMessage>
            <ErrorCode>601</ErrorCode>
      </Error>
</Response>
```

Example Request (602):

```
<?xml version='1.0'?>
<Request>
      <Authentication>
            <Username><![CDATA[MyUsername]]></Username>
            <Password><![CDATA[MyPassword]]></Password>
      </Authentication>

      <AccountDetails>
            <RequestStatus>
                  <Ticket></Ticket>
            </RequestStatus>
      </AccountDetails>
</Request>
```

Example Response (602):

```
<?xml version='1.0'?>
<Response>
      <Error>
            <ErrorMessage>Your xml is setting a null value or some data is too
      long, check that each tag contains a valid value. NOTE - max message
      text character length is 1600, if you are sure you need to send a
      message longer than this please contact the txttools technical team on
      techies@txttools.co.uk</ErrorMessage>
            <ErrorCode>602</ErrorCode>
      </Error>
</Response>
```

Example Request (604):

```
<?xml version='1.0'?>
<Request>
      <Authentication>
            <Username><![CDATA[MyUsername]]>
      </Authentication>
</Request>
```

Example Response (604):

```
<?xml version='1.0'?>
<Response>
      <Error>
            <ErrorMessage>The element type "Username" must be terminated by the
      matching end-tag "</Username>".</ErrorMessage>
            <ErrorCode>604</ErrorCode>
      </Error>
</Response>
```

XML Push API for Messaging

What is the XML Push API?

ConnectTxt offers customers the ability to provide a URI which can accept delivery of message status updates and inbound SMS messages. Any messages or status updates received by ConnectTxt are automatically 'pushed' to this URI without any intervention required from the customer's system. This allows developers to build applications without the added complexity inherent in needing to continuously poll the ConnectTxt server for status information and messages.

How to use the API

Customers who wish to implement this interface must provide ConnectTxt with a URI for the ConnectTxt system to connect to. This URI should accept three parameters (via POST, not GET):

Parameter	Description
u	A username (specified by the customer), used to authenticate the connection.
p	A password (specified by the customer), used to authenticate the connection.
x	The XML being sent by the ConnectTxt system.

A few notes about the XML push API:

1. The XML payload structure follows the same schema as shown for the messaging API earlier in this document (but contains only inbound messages and status updates).

2. SSL encryption is supported on the push API, and is a recommended transmission method, to ensure secure use of the system.

3. Please ensure that your firewall is correctly configured to allow incoming TCP connections to this URI (typically on port 80 for HTTP or port 443 for SSL) .

4. The structure for pushed inbound messages differs slightly from that provided when requesting messages, to help indicate which account/number the message was sent to. Please see below:

InboundMessage Response Structure in XML Push:

Element	Description	Required
InboundMessage	Top level element for this structure.	**YES**
Ticket	The internal ticket for this message.	**YES**
MessageText	The text of the message.	**YES**
Phone	The phone number of the sender.	**YES**
Date	A UTC timestamp in seconds showing when the message was received at ConnectTxt.	**YES**
Destination	The phone number that the message was sent to.	**YES**
DestinationAccount	The account username that the message was pushed from.	**YES**

Example Request:

```
<InboundMessage>
      <Ticket>1278</Ticket>
      <MessageText><![CDATA[Example]]></MessageText>
      <Phone><![CDATA[+4471234567890]]></Phone>
      <Date>1234567890</Date>
      <Destination>
            <![CDATA[+449876543210]]>
      </Destination>
      <DestinationAccount>
            <![CDATA[GregJPreece]]>
      </DestinationAccount>
</InboundMessage>
```

Version History

2nd May 2012 - ConnectTxt 6.8.1

Added new milliseconds-based search on inbound messages.
Updated scheduled messages to use new ScheduleTimeUTCSecs tag over MessageDate
Updated product name from txttools to ConnectTxt

7th Feb 2012

Company rebranding to Blackboard.

13th October 2011 - txttools 6.7

Clarified error codes and added new 600 code.

28th March 2011 - txttools 6.6.2

Added version history to document
Added hint regarding Transfer-Encoding header for HTTP 1.1 clients
Re-added missing error code info
Added new inbox counts documentation
Added reporting levels to outbound messages
Updated account detail requests to new version