



## **Using the Tizra Publisher REST API**

Version: 1.0  
Issued: January 1, 2009

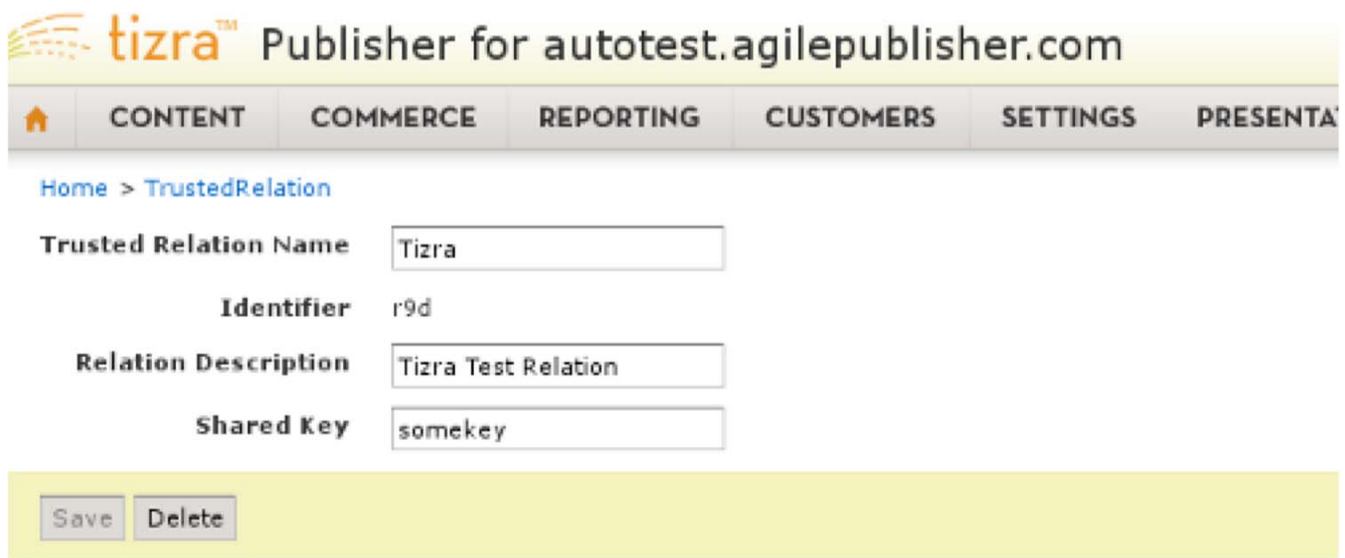
# Overview of the REST API

Tizra Publisher provides a REST API to allow clients to programmatically access the system through their own applications, rather than requiring that every interaction be performed through the web user interface. For example, a client may wish to write an application that automatically creates new user accounts, adds or deletes licenses, etc.

Clients wishing to use the REST API must create a Trusted Relation in the administrative user interface, with a name and a shared key. This name and key are used to authenticate interactions via the REST API.

## *Preparing for use*

To create a Trusted Relation for your site, log in to the administrative interface. Go to the “Commerce” tab, then the “Trusted Relations” link. Choose the “Create New” button. Fill in your new relation's name, description (optional), and shared key. Choose the “Create” button. When taken to the information page for the newly created relation, be sure to note the relation's ID, as that string will be required in its use.



The screenshot shows the administrative interface for Tizra Publisher. The header includes the Tizra logo and the site name 'Publisher for autotest.agilepublisher.com'. A navigation bar contains tabs for 'CONTENT', 'COMMERCE', 'REPORTING', 'CUSTOMERS', 'SETTINGS', and 'PRESENTA'. Below the navigation bar, the breadcrumb 'Home > TrustedRelation' is visible. The main form area contains four fields: 'Trusted Relation Name' with the value 'Tizra', 'Identifier' with the value 'r9d', 'Relation Description' with the value 'Tizra Test Relation', and 'Shared Key' with the value 'somekey'. At the bottom of the form are 'Save' and 'Delete' buttons.

[Terms of Service](#) | [About Tizra](#)

In this tutorial, we will assume that you have created Trusted Relation “Tizra” with identifier “r9d” and shared key “somekey”, for the site autotest.agilepublisher.com.

## ***Formatting a REST request***

Most requests to the REST server will contain an XML message body, which must conform to the trustmessage DTD:

```
<!ELEMENT trustmessage (parameter*)>
<!ELEMENT parameter (name,value)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

In short, the XML message body must have a “trustmessage” root element, which will have zero or more “parameter” child elements. Each “parameter” element will have a “name” and “value” child element, containing the name and the value of the parameter in question, respectively. Example XML documents will be given for each REST Trusted Relation command.

Responses from the REST server will similarly contain an XML message body, using the same DTD.

## ***The authentication flow***

To execute a useful REST API request, first send an authentication request, including Trusted Relation name and shared key, to the REST server. The REST server will return an authentication token (or an error, if the authentication information is not correct). Save the token returned. In future requests, set the “Authorization” HTTP header on each request, using the given token as its value. The token will be valid for five minutes after the most recent request (i.e., the user may be idle for five minutes before it expires).

## **The REST API**

### ***Requesting an authorization token***

URL	/trust/<relationId>/authorization
Method	POST
Parameters	authentication, authenticationdate
Action	Returns authentication token

Your web client should send a POST request, with some XML in the message body specifying

authentication parameters. The first required parameter is “authenticationdate,” the current time in milliseconds (GMT). Note that because an authenticationdate more than five minutes old will be invalid, client machines must be synchronized to the correct time. We recommend using a tool such as ntp for this purpose. The second required parameter is “authentication,” which is an MD5 hexed string:

```
/trust/<relationId>/authorization<sharedKey><authenticationDate>
```

For example, an unhexed authentication string might look like

```
/trust/r9d/authorizationsomekey1230841145270
```

And the hexed version might look like

```
9c129c8b7c11fa2ce9787013f0eb8941
```

**Sample request**

URL	/trust/9rd/authorization
Method	POST
Headers	
XML:	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "-//Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/trustmessage.dtd"&gt;  &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;authenticationdate&lt;/name&gt;     &lt;value&gt;1230841714979&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;authentication&lt;/name&gt;</pre>

	<pre> &lt;value&gt;918fb4d592f16c5cbe28d7ecc6 cabdd5&lt;/value&gt;   &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>
--	--

The REST server will respond with an XML document in similar format, including an “authorization” parameter. The value of that parameter will be an authorization token which can be used in future requests to the server. This token will be valid for ten minutes from its creation or from its most recent use.

If there has been an error, the error message will be specified in the “errorMessage” parameter, and the HTTP return code will be set to SC\_FORBIDDEN (403).

### **Sample Response**

Return Type	SC_OK (200)
Headers	
XML	<pre> &lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/tr ustmessage.dtd"&gt; &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;authorization&lt;/name&gt;     &lt;value&gt;/trust/r9d/session/1274476 41230842541705&lt;/value&gt;   &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>

## Creating a user with the REST API

URL	/trust/<relationId>/users
Method	POST
Parameters	Username, password (optional), institutional (optional)
Action	Creates a new user

Your web client should send a POST request to the specified URL, and include the username to create. You may specify a password; if you don't, one will be generated for you and returned in the XML response (as parameter "password"). By default, the user will be created as an individual account, but you can specify that "institutional" is "true" to create an institutional account instead.

A username should be an email address. Errors will be returned on inappropriate usernames, passwords that are less than 4 characters long, or preexisting usernames.

### **Sample request**

URL	/trust/9rp/users
Method	POST
Headers	Authentication=9c129c8b7c11fa2ce9787013f0eb8941
XML:	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/tr ustmessage.dtd"&gt; &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;username&lt;/name&gt;     &lt;value&gt;<a href="mailto:testuser@tizra.com">testuser@tizra.com</a>&lt;/value&gt;   &lt;/parameter&gt;</pre>

	<pre> &lt;parameter&gt;   &lt;name&gt;userId&lt;/name&gt;   &lt;value&gt;r9e&lt;/value&gt; &lt;/parameter&gt; &lt;parameter&gt;   &lt;name&gt;password&lt;/name&gt;   &lt;value&gt;samplepassword&lt;/value&gt; &lt;/parameter&gt; &lt;parameter&gt;   &lt;name&gt;institutional&lt;/name&gt;   &lt;value&gt;&gt;true&lt;/value&gt; &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>
--	---

The response will include parameters for the new username, password, userId, and account type (institutional or individual). Errors, if any, will be specified in the errorMessage parameter. Expected return type is SC\_CREATED (201), but will be SC\_BAD\_REQUEST on error (400). Sample response:

**Sample Response**

Return Type	SC_CREATED (201)
Headers	
XML	<pre> &lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/tr ustmessage.dtd"&gt;  &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;username&lt;/name&gt; </pre>

	<pre> &lt;value&gt;testuser@tizra.com&lt;/value&gt; &lt;/parameter&gt; &lt;parameter&gt;   &lt;name&gt;password&lt;/name&gt;   &lt;value&gt;samplepassword&lt;/value&gt; &lt;/parameter&gt; &lt;parameter&gt;   &lt;name&gt;userId&lt;/name&gt;   &lt;value&gt;r9l&lt;/value&gt; &lt;/parameter&gt; &lt;parameter&gt;   &lt;name&gt;accountType&lt;/name&gt;   &lt;value&gt;institutional&lt;/value&gt; &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>
--	---

## ***Retrieve user information with the REST API***

URL	/trust/<relationId>/users/<userId>
Method	GET
Parameters	
Action	Retrieve information for an existing user account

Request information for an existing user, including username, activation status (“active,” “pending,” or “canceled”), account type (“institutional” or “individual”), and MetaTag values. The user's password will not be returned. Because no parameters are required, this command does not require an accompanying XML trustmessage; the request body may be empty. Errors, if any, will be specified in the errorMessage parameter. An error will be returned if a non-existent account is specified. Expected return type is SC\_OK (200), but will be SC\_BAD\_REQUEST on error (400).

### **Sample request**

URL	/trust/9rp/users/9jd
Method	GET
Headers	Authentication=9c129c8b7c11fa2ce9787013f0eb8941
XML:	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "-//Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/trustmessage.dtd"&gt; &lt;trustmessage&gt;&lt;/trustmessage&gt;</pre>

### **Sample Response**

Return Type	SC_OK (200)
Headers	
XML	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "-//Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/trustmessage.dtd"&gt;  &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;username&lt;/name&gt;      &lt;value&gt;testuser2@<a href="http://www.tizra.com">tizra.com</a>&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;status&lt;/name&gt;     &lt;value&gt;canceled&lt;/value&gt;   &lt;/parameter&gt;</pre>

	<pre> &lt;parameter&gt;   &lt;name&gt;accountType&lt;/name&gt;   &lt;value&gt;individual&lt;/value&gt; &lt;/parameter&gt; &lt;parameter&gt;   &lt;name&gt;FirstName&lt;/name&gt;   &lt;value&gt;John&lt;/value&gt; &lt;/parameter&gt; &lt;parameter&gt;   &lt;name&gt;LastName&lt;/name&gt;   &lt;value&gt;Smith&lt;/value&gt; &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>
--	--

## ***Editing user information with the REST API***

URL	/trust/<relationId>/users/<userId>
Method	PUT
Parameters	username (optional), password (optional), MetaTags to change (optional), active (optional)
Action	Modifies existing user account

Specify a new username, new password, new active status, and/or new values for MetaTags for a user. Note that the user is specified based on the userId in the request URL; the “username” parameter is intended only for changing the username to a new value. “Active” may be “true” or “false” to make a user active or inactive.

The response will include the username, account activity status, and current values of all MetaTags set for this user. Note that information that was not changed will also be returned.

Errors will be returned if the “username” parameter is set and the given user has multiple username

credentials (so that it is not clear which to change), if the new username is invalid, if the new username is already in use, or if the new password is shorter than 4 characters. Unrecognized parameters (e.g. unrecognized MetaTag names) will be silently discarded.

**Sample request**

URL	/trust/9rp/users/9jd
Method	PUT
Headers	Authentication=9c129c8b7c11fa2ce9787013f0eb8941
XML:	<pre> &lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "-//Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "<a href="http://www.tizra.com/agilepdf/trustmessage.dtd">http://www.tizra.com/agilepdf/trustmessage.dtd</a>"&gt;  &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;username&lt;/name&gt;  &lt;value&gt;testuser2@<a href="http://www.tizra.com">tizra.com</a>&lt;/value&gt; &gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;active&lt;/name&gt;     &lt;value&gt;&gt;false&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;FirstName&lt;/name&gt;     &lt;value&gt;John&lt;/value&gt;   &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>

## Sample Response

Return Type	SC_OK (200)
Headers	
XML	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/tr ustmessage.dtd"&gt; &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;username&lt;/name&gt;     &lt;value&gt;testuser2@tizra.com&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;status&lt;/name&gt;     &lt;value&gt;canceled&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;accountType&lt;/name&gt;     &lt;value&gt;individual&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;FirstName&lt;/name&gt;     &lt;value&gt;John&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;LastName&lt;/name&gt;     &lt;value&gt;Smith&lt;/value&gt;   &lt;/parameter&gt;</pre>

	</trustmessage>
--	-----------------

## **Delete user account with the REST API**

URL	/trust/<relationId>/users/<userId>
Method	DELETE
Parameters	
Action	Delete an existing user account

Delete an existing user account. Because no parameters are required, this command does not require an accompanying XML trustmessage; the request body may be empty. An error will be returned if the account does not exist. Error messages will be set in the errorMessage parameter. Expected return code is SC\_OK (200) but on error will be SC\_BAD\_REQUEST (400).

### **Sample request**

URL	/trust/9rp/users/9jd
Method	DELETE
Headers	Authentication=9c129c8b7c11fa2ce9787013f0eb8941
XML:	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/tr ustmessage.dtd"&gt;  &lt;trustmessage&gt;&lt;/trustmessage&gt;</pre>

### **Sample Response**

Return Type	SC_OK (200)
-------------	-------------

Headers	
XML	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "-//Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/trustmessage.dtd"&gt;  &lt;trustmessage&gt;&lt;/trustmessage&gt;</pre>

## ***Add license to a user's account with the REST API***

URL	/trust/<relationId>/licenses/<userId>
Method	POST
Parameters	offerId
Action	Add a license to an existing user's account

Add a license for the specified offer (“offerId”) to a user's account. The response will return the ID of the newly-created license (“licenseId”). An error will be returned if the user account does not exist or the offer does not exist. Error messages will be set in the errorMessage parameter. Expected return code is SC\_OK (200) but on error will be SC\_BAD\_REQUEST (400).

### **Sample request**

URL	/trust/9rp/licenses/9jd
Method	POST
Headers	Authentication=9c129c8b7c11fa2ce9787013f0eb8941
XML:	<pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "-//Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/trustmessage.dtd"&gt;</pre>

	<pre> <a href="#">ustmessage.dtd"&gt;</a>  &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;offerId&lt;/name&gt;     &lt;value&gt;anf&lt;/value&gt;   &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>
--	---

**Sample Response**

Return Type	SC_OK (200)
Headers	
XML	<pre> &lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" <a href="http://www.tizra.com/agilepdf/trustmessage.dtd">"http://www.tizra.com/agilepdf/trustmessage.dtd"&gt;</a>  &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;licenseId&lt;/name&gt;     &lt;value&gt;r9j&lt;/value&gt;   &lt;/parameter&gt; &lt;/trustmessage&gt; </pre>

***Log on a user (create a new session) with the REST API***

URL	/trust/<relationId>/sessions
Method	POST
Parameters	Username, redirecturl
Action	Create a new session for a given user

Create a new session for the specified user. The response will include a “Location” header with an appropriate URL for redirection. This URL will include an authentication token. The requesting web client should redirect to this URL, after which the specified user will be logged in. If the web client does not redirect to the “Location” URL, the user will not be logged in.

An error will be returned if the user account does not exist. Error messages will be set in the errorMessage parameter. Expected return code is SC\_CREATED (201) but on error will be SC\_BAD\_REQUEST (400).

### **Sample request**

URL	/trust/9rp/sessions
Method	POST
Headers	Authentication=9c129c8b7c11fa2ce9787013f0eb8941
XML:	<pre> &lt;?xml version="1.0"?&gt; &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/tr ustmessage.dtd"&gt;  &lt;trustmessage&gt;   &lt;parameter&gt;     &lt;name&gt;username&lt;/name&gt;  &lt;value&gt;testuser@tizra.com&lt;/value&gt;   &lt;/parameter&gt;   &lt;parameter&gt;     &lt;name&gt;redirecturl&lt;/name&gt; </pre>

	<pre>&lt;value&gt;http://autotest.agilepubli sher.com/&lt;/value&gt;  &lt;/parameter&gt;  &lt;/trustmessage&gt;</pre>
--	---

### **Sample Response**

Return Type	SC_CREATED (201)
Headers	Location=http://autotest.agilepublis her.com:80/authcallback/9p8/inter nal?authToken=2409958312312795455 30&target=%2Fview%2F9p8%2Fdefault 1
XML	<pre>&lt;?xml version="1.0"?&gt;  &lt;!DOCTYPE trustmessage PUBLIC "- //Tizra//DTD AgilePDF Trustmessage XML Catalog V1.0//EN" "http://www.tizra.com/agilepdf/tr ustmessage.dtd"&gt;  &lt;trustmessage&gt;&lt;/trustmessage&gt;</pre>

## **Sample code: Using the API in Java**

Below is source code for a utilities class, SampleRESTUtilities. This class contains methods to request an authorization token, create a new user via the REST API using that token, and parse an XML response message body. It uses the org.apache.commons HttpClient package to make HTTP requests. This class is intended as sample code only, not for production use.

```
package com.tizrapublisher.trust.util;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.io.ByteArrayInputStream;
```

```
import java.io.InputStream;
import javax.servlet.http.HttpServletResponse;

import java.util.Calendar;
import java.util.Formatter;
import java.util.HashMap;
import java.util.Map;

import org.apache.xml.resolver.tools.CatalogResolver;
import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.methods.*;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.Text;
import org.jdom.input.SAXBuilder;
```

```
/**
 * Util class for TrustedRelation REST convenience methods.
 * @author Jessica P. Hekman
 */
```

```
public class SampleRESTUtilities {

    // table to convert a nibble to a hex character

    private static char[] hexChar = {'0', '1', '2', '3',
        '4', '5', '6', '7',
        '8', '9', 'a', 'b',
```

```

        'c', 'd', 'e', 'f');
static MessageDigest md = null;
static {
    try {
        md = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException e) {
        // can't happen? (famous last words)
    }
}

static public String requestAuthorizationUrl(String relationId, String sharedKey, String hostname,
String port) throws Exception {
    String url = "trust/" + relationId + "/authorization";

    HttpClient client = new HttpClient();
    PostMethod post = new PostMethod("http://" + hostname + ":" + port + "/" + url);
    HashMap<String,String> parameters = new HashMap<String,String>();
    long millis = Calendar.getInstance().getTimeInMillis();

    // Hex the authentication string
    String toHex = "/" + url + sharedKey + millis;
    byte[] md5Bytes = md.digest(toHex.getBytes());
    StringBuffer hexBuf = new StringBuffer();
    Formatter f = new Formatter(hexBuf);
    for (byte b : md5Bytes) {
        f.format("%02x", b);
    }
    String hexed = hexBuf.toString();

    parameters.put("authentication",hexed);
    parameters.put("authenticationdate",new Long(millis).toString());
}

```

```

post.setRequestEntity(SampleRESTUtilities.generateRequestInputStream(parameters));
int responseCode = client.executeMethod(post);
if (responseCode == HttpServletResponse.SC_FORBIDDEN) {
    Map<String,String> requestParameters =
SampleRESTUtilities.parseXmlContent(post.getResponseBodyAsStream());
    throw new Exception ("Authorization failed: " + requestParameters.get("errorMessage"));
}

Map<String,String> responseParameters =
SampleRESTUtilities.parseXmlContent(post.getResponseBodyAsStream());
String authorization = responseParameters.get("authorization");
if (authorization == null) {
    throw new Exception ("No 'Authorization' header in response.");
}
return authorization;
}

/**
 * Parse an XML REST document.
 * @param xmlContent the XML REST document to parse
 */
static public Map<String,String> parseXmlContent(InputStream xmlContent) throws Exception {

SAXBuilder builder = new SAXBuilder(true);
builder.setEntityResolver(new CatalogResolver());
builder.setIgnoringElementContentWhitespace(true);
Document xmlDoc = builder.build(xmlContent);
Element rootElement = xmlDoc.getRootElement();

HashMap<String,String> requestParameters = new HashMap<String,String>();
for (Object oneObject : rootElement.getChildren()) {

```

```

if (oneObject instanceof Element) {
    Element oneElement = (Element) oneObject;
    if ("parameter".equals(oneElement.getName())) {
        String name = null;
        String value = null;
        for (Object innerObject : oneElement.getChildren()) {
            if (innerObject instanceof Element) {
                Element innerElement = (Element) innerObject;
                if ("name".equals(innerElement.getName())) {
                    name = SampleRESTUtilities.getXMLElementInnerText(innerElement).trim();
                } else if ("value".equals(innerElement.getName())) {
                    value = SampleRESTUtilities.getXMLElementInnerText(innerElement).trim();
                }
            }
            requestParameters.put(name, value);
        }
    }
}
return requestParameters;
}

```

*/\* Create a user via REST. \*/*

```

public static String doCreateUserCommand(String relationId, String newUsername, String
newPassword, String newUserType, String sharedKey, String hostname, String hostport) throws
Exception {

```

```

    if (newUsername == null || newPassword == null) {
        throw new Exception ("Please specify username and password.");
    }

```

```

String authorizationUrl = SampleRESTUtilities.requestAuthorizationUrl(relationId, sharedKey,

```

```

hostname, hostport);
    assert authorizationUrl != null : "Expected non-null authorization response from REST server.";

    HashMap<String,String> parameters = new HashMap<String,String>();
    parameters.put("username", newUsername);
    parameters.put("password", newPassword);
    if ("institutional".equals(newUserType)) {
        parameters.put("institutional", "true");
    }
    String url = "trust/" + relationId + "/users";
    HttpClient client = new HttpClient();
    PostMethod post = new PostMethod("http://" + hostname + ":" + hostport + "/" + url);
    post.setRequestHeader("Authorization", authorizationUrl);
    post.setRequestEntity(SampleRESTUtilities.generateRequestInputStream(parameters));
    int responseCode = client.executeMethod(post);

    // check error code
    if (responseCode != HttpServletResponse.SC_CREATED) {
        return "failed";
    }
    return "success";
}

/*
 * Create an HttpClient RequestEntity containing an appropriately-formatted request for the REST
server.
 * @param parameters HashMap of parameters to pass to the REST server
 * @returns RequestEntity containing formatted request in XML
 */
static public RequestEntity generateRequestInputStream (HashMap<String,String> parameters)
throws Exception {

```

```
StringBuffer content = new StringBuffer();
```

```
    content.append("<?xml version=\"1.0\"?><!DOCTYPE trustmessage PUBLIC \"-//Tizra//DTD  
AgilePDF Trustmessage XML Catalog V1.0/EN\"  
\"http://www.tizra.com/agilepdf/trustmessage.dtd\">\n<trustmessage>");
```

```
    if (parameters != null) {
```

```
        for (String parameter : parameters.keySet()) {
```

```
            content.append("<parameter><name>");
```

```
            content.append(parameter);
```

```
            content.append("</name>\n<value>");
```

```
            if (parameters.get(parameter) != null) {
```

```
                content.append(parameters.get(parameter));
```

```
            }
```

```
            content.append("</value>\n</parameter>");
```

```
        }
```

```
    }
```

```
    content.append("</trustmessage>\n");
```

```
    byte bytes[] = content.toString().getBytes();
```

```
    return (new InputStreamRequestEntity(new ByteArrayInputStream(bytes)));
```

```
}
```

```
/**
```

```
 * Extract the content text from an XML Element, accomodate
```

```
 * for empty tagging (i.e. <oneTag />) and still allow for
```

```
 * empty result to be returned.
```

```
 * @param oneElement the element to extract inner text for.
```

```
 * @return the element's inner text
```

```
 */
```

```
public static String getXMLElementInnerText( Element oneElement ) {
```

```
    if ( oneElement.getContentSize() > 0 ) {
```

```
Object innerContent = oneElement.getContent(0);
if ( innerContent instanceof Text ) {
    return ((Text)innerContent).getText();
}
}
return "";
}
}
```