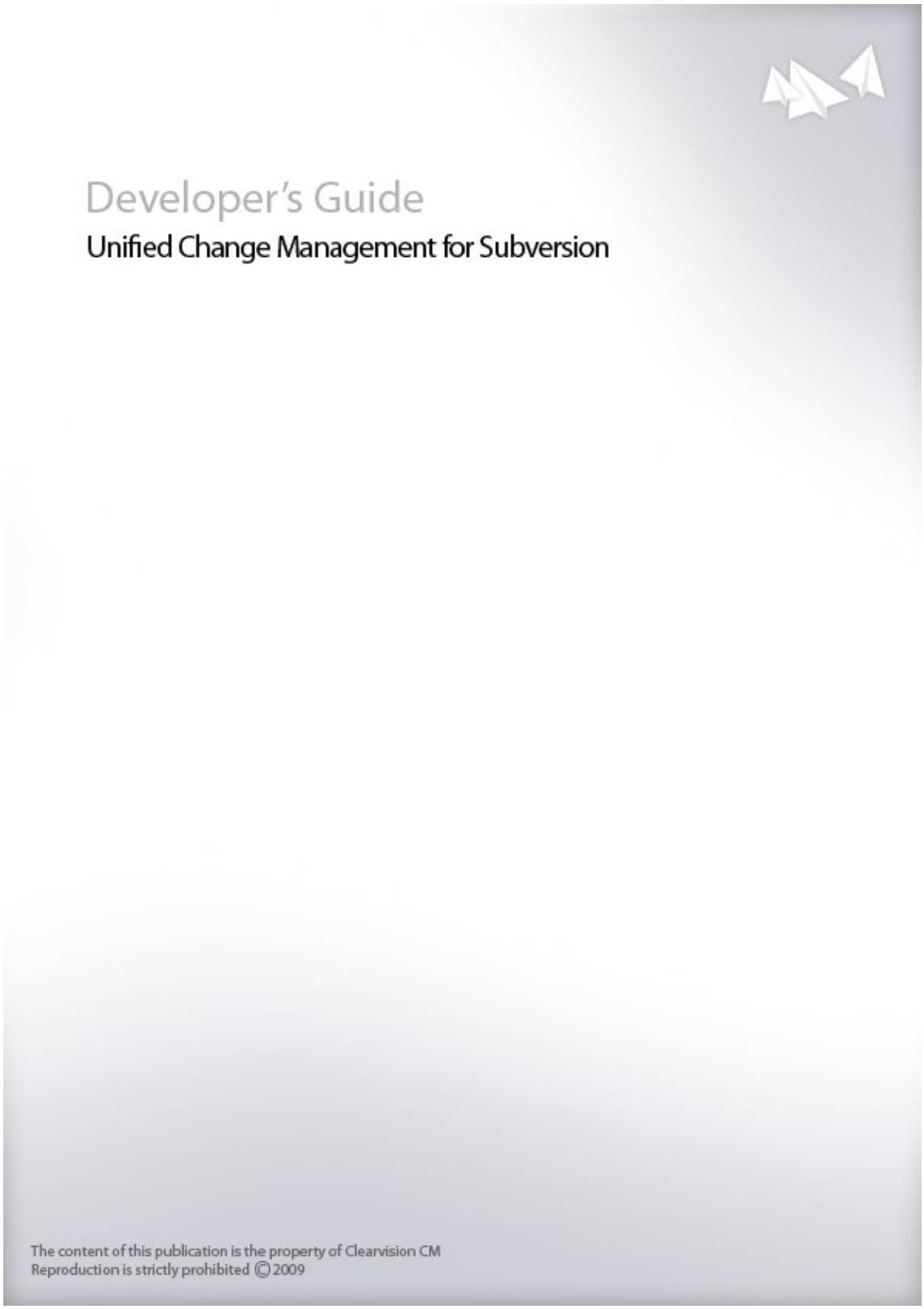

Contents

1 Introduction

This guide provides information on the Application Programming Interface (API) provided by UCM4SVN. It documents the API functions currently available and provides information on how to access the API.

The main interface to UCM4SVN is the web interface provided and documented in the UCM4SVN User Guide. As an ordinary user of UCM4SVN, you should not need to use the public API.

The API is provided specifically for users wishing to script day-to-day tasks or integrate UCM4SVN into their own applications. Please note that currently only a small subset of UCM4SVN functionality is available through the API.

1.1 UCM4SVN API

UCM4SVN provides a public API, which follows the principles of RESTful Web Services (see REST or Representational State Transfer on Wikipedia for an overview). The API can be accessed using the base URI `/api/`, e.g. for a UCM4SVN server at `http://my.company.com:54323`, the API lives at `http://my.company.com:54323/api/`.

Each API method has its own URI location (e.g. `http://my.company.com:543232/api/project/` for project-related methods) and HTTP request type. Getter methods such as getting a list of projects or getting a list of users use the HTTP GET method. Creating new items involves an HTTP POST. And existing items are changed using a PUT request. The last remaining request type DELETE is for deleting items from a collection. However, there are currently no DELETE requests in the UCM4SVN API.

Before an API function can be called, the user needs to log in to the API with the `login` method. Logging in returns a token, which needs to be passed to subsequent calls to the API. Every call other than login requires a valid token, or the request will be rejected.

Tokens time out after ten minutes of inactivity. However, you can also release a token earlier using a call to `logout`.

Please note that both `login` and `logout` are POST requests. Whilst this is consistent with REST for `login` (logging in creates a new token, which makes it a POST request), it could be argued that `logout` should be implemented as a DELETE. However, it was decided to make an exception for logout and implement it as a POST in order to make it more accessible to a wider range of users (GET and POST requests are easier to create in a large range of network-programming libraries than DELETE).

All API calls return a response data structure in XML format. For example, a call to `login` returns the following XML result:-

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:token>51f46556061</cv:token>¶
</cv:response>¶
```

The basic format is the same for all calls. Every call returns an XML document containing a `cv:response` top-level element, with the first child being `cv:status`. The status element indicates whether an operation was successful. If the call was successful, the attribute "cv:success is "true". For unsuccessful calls, the error message is provided in the status element and the attribute "cv:success" is set to "false".

Please note that sufficient permissions are required for an API call to be successful. Administrators and Project Managers have access to most API calls. However, Developers must be a member of a project in order to be able to access data related to a project. Also, the normal role restrictions apply, e.g. a Developer cannot create baselines through the API.

1.2 Accessing the API from Programming Languages

As access to the API is provided through HTTP requests, the API can be used from any programming language that provides access to network sockets. However, access to the API becomes even easier with most modern programming languages, which provide an HTTP layer on top of simple TCP/IP network access.

This section provides examples on how to access the UCM4SVN API from Java and Python. However, the concepts shown can easily be mapped onto other programming languages. Please note that examples are designed to show the core functionality and do not show the necessary error handling required for robust applications. This is done deliberately to keep the examples clean and easy to follow.

1.2.1 Java Example

When interfacing to UCM4SVN from Java, simple packages such as `java.net` can be used to interface to network resources. The XML result returned from UCM4SVN can be parsed into a DOM (Document Object Model) and accessed via the usual access functions.

The example below uses `java.net` and `org.w3c.dom`. However, you may find it easier to use XPath processors or other XML libraries such as Apache Xerces. XSLT processors such as Xalan in combination with XSL scripts may also be helpful, depending on your application.

```
import java.io.OutputStreamWriter;¶
import java.net.URL;¶
import java.net.URLConnection;¶
import java.net.URLEncoder;¶
import java.util.ArrayList;¶
import javax.xml.parsers.DocumentBuilder;¶
import javax.xml.parsers.DocumentBuilderFactory;¶
import org.w3c.dom.Document;¶
import org.w3c.dom.NodeList;¶
¶
¶
public class Ucm4svnAccess {¶
··· public String login(String username, String password) {¶
····· String token = null;¶
····· try {¶
····· // Create POST parameters¶
····· String params;¶
····· params = URLEncoder.encode("username", "UTF-8") + "=" + ¶
·····  URLEncoder.encode("admin", "UTF-8") + "&" + ¶
·····  URLEncoder.encode("password", "UTF-8") + "=" + ¶
·····  URLEncoder.encode("admin", "UTF-8");¶
··· ¶
····· URL url = new URL("http://192.168.1.27:54323/api/login/");¶
····· URLConnection conn = url.openConnection();¶
····· conn.setDoOutput(true);¶
····· ¶
····· // POST the arguments¶
```

```
·········· OutputStreamWriter wr = new
OutputStreamWriter(conn.getOutputStream());¶
·········· wr.write(params);¶
·········· wr.flush();¶
¶
·········· // Get the response and read into XML DOM¶
·········· DocumentBuilderFactory builderFactory =
DocumentBuilderFactory.newInstance();¶
·········· DocumentBuilder builder =
builderFactory.newDocumentBuilder();¶
¶
·········· Document doc = builder.parse(conn.getInputStream());¶
·········· token_element =
doc.getElementsByTagName("cv:token").item(0);¶
·········· token =
token_element.getChildNodes().item(0).getNodeValue();¶
·········· ¶
·········· wr.close();¶
····· } catch (Exception e) {¶
·········· System.out.println("Error occurred: " + e.getMessage());¶
····· }¶
····· return token;¶
··· }¶
··· ¶
··· public String[] getComponentNames(String token, String projectName)
{¶
····· ArrayList<String> componentNamesList = new ArrayList<String>();¶
····· try {¶
·········· // Create GET parameters¶
·········· String params;¶
·········· params = URLEncoder.encode("token", "UTF-8") + "=" + ¶
·····················URLEncoder.encode(token, "UTF-8") + "&" + ¶
·····················URLEncoder.encode("project_name", "UTF-8") + "=" +
¶
·····················URLEncoder.encode(projectName, "UTF-8");¶
··· ¶
·········· URL url = new URL("http://192.168.1.27:54323/api/component?"
+ params);¶
·········· URLConnection conn = url.openConnection();¶
·········· conn.setDoOutput(true);¶
·········· ¶
·········· // Get the response and read into XML DOM¶
·········· DocumentBuilderFactory builderFactory =
DocumentBuilderFactory.newInstance();¶
·········· DocumentBuilder builder =
builderFactory.newDocumentBuilder();¶
¶
·········· Document doc = builder.parse(conn.getInputStream());¶
·········· ¶
·········· NodeList nodeList = doc.getElementsByTagName("cv:name");¶
·········· for (int i = 0; i < nodeList.getLength(); i++) {¶
·············· String item =
nodeList.item(i).getChildNodes().item(0).getNodeValue();¶
·············· componentNamesList.add(item);¶
```

```
···········}¶
···········¶
·······} catch (Exception e) {¶
···········System.out.println("Error occurred: " + e.getMessage());¶
·······}¶
·······String[] componentNames = new
String[componentNamesList.size()];¶
·······return (String[])componentNamesList.toArray(componentNames);¶
···}¶
¶
···public static void main(String argv[]) {¶
·······Ucm4svnAccess ucm4svn = new Ucm4svnAccess();¶
·······System.out.println("Starting");¶
·······String token = ucm4svn.login("admin", "admin");¶
·······System.out.println("Logged in with token " + token);¶
·······for (String componentName: ucm4svn.getComponentNames(token,
"Ferrari F50")) {¶
···········System.out.println("Component: " + componentName);¶
·······}¶
·······¶
···}¶
}¶
```

1.2.2 Python Example

The Python `urllib` library makes creating GET and POST requests relatively easy. The XML response can be parsed into a DOM (Document Object Model), from which information can be retrieved using the usual DOM access methods.

PUT requests in Python can be created using the `httplib` API.

Example:-

```
import urllib¶
import xml.dom.minidom¶
¶
url = "http://ucm4svn.mycompany.com:54323"¶
¶
# Create a POST request to login to UCM4SVN¶
params = urllib.urlencode({'username': 'admin', 'password': 'admin'})¶
f = urllib.urlopen(url + "/api/login/", params)¶
¶
doc = xml.dom.minidom.parseString(f.read())¶
status_element = doc.getElementsByTagName("cv:status")[0]¶
status = status_element.attributes['cv:success'].value¶
token = tokens =
doc.getElementsByTagName("cv:token")[0].firstChild.nodeValue¶
¶
print "Login returned status '%s', token '%s'" % (status, token)¶
¶
# Get a list of components for project project1¶
f = urllib.urlopen(url +
```

```
"/api/component?token=%s&project_name=Ferrari%%20F50" % (token))¶
doc = xml.dom.minidom.parseString(f.read())¶
¶
component_names = []¶
for component_name_element in doc.getElementsByTagName("cv:name"):¶
··· component_names.append(component_name_element.firstChild.nodeValue)¶
··· ¶
print "Component list: %s" % str(component_names)¶
```

2 API Reference

2.1 Login

Description

HTTP Request Type	POST
URL	/api/login/

Login to UCM4SVN and generate a new token. All API functions other than `login` need a valid token to function. Please note that tokens expire if they are not used for more than ten minutes, or if they are released with a call to `logout`.

Arguments

username	User name.
password	Password.

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:token>2612e21c252</cv:token>¶
</cv:response>¶
```

2.2 Logout

Description

HTTP Request Type	POST
URL	/api/logout/

Release a login token.

Arguments

token	Token to be released.

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
</cv:response>¶
```

2.3 Get Projects

Description

HTTP Request Type	GET
URL	/api/project/

Get a list of projects from UCM4SVN.

Arguments

token	Login token

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:projects>¶
····· <cv:project>¶
····· <cv:id>1</cv:id>¶
····· <cv:name>Focus 1.6LX</cv:name>¶
····· </cv:project>¶
····· <cv:project>¶
····· <cv:id>2</cv:id>¶
····· <cv:name>Focus 2.5GT</cv:name>¶
····· </cv:project>¶
··· </cv:projects>¶
</cv:response>¶
```

2.4 Get Components

Description

HTTP Request Type	GET
URL	/api/component/

|| URL || /api/project/<project_id>/component/ ||

Get a list of components. The list of components can be narrowed down to components that are part of a particular project.

Arguments

token	Login token

Response

If the request was for components for a particular project, the `cv:access` indicates whether a component is read-only or read-write for the project. If the request was for a complete list of components, there will be no {cv:access element as this is a property of the relationship between a project and a component. It is only when you view a component within the context of a project that you can indicate whether or not the component is writeable for that project.

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:components>¶
····· <cv:component>¶
····· <cv:id>2</cv:id>¶
····· <cv:name>engine</cv:name>¶
····· <cv:access>read-write</cv:access>¶
····· </cv:component>¶
····· <cv:component>¶
····· <cv:id>8</cv:id>¶
····· <cv:name>gearbox</cv:name>¶
····· <cv:access>read-only</cv:access>¶
····· </cv:component>¶
··· </cv:components>¶
</cv:response>¶
```

2.5 Get Recommended Baseline

Description

HTTP Request Type	GET
URL	/api/project/<project_id>/component/<component_id>/recommended_baseline/
URL	/api/project/<project_id>/recommended_baseline/

Get the recommended baseline for a particular component in a particular project or all the components in a project.

Arguments

token	Login token

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:recommended_baselines>¶
····· <cv:recommended_baseline>¶
····· <cv:component>¶
····· <cv:id>3</cv:id>¶
····· <cv:name>engine</cv:name>¶
```

```
·········· </cv:component>¶
·········· <cv:baseline>¶
·············· <cv:id>8</cv:id>¶
·············· <cv:name>base3</cv:name>¶
·········· </cv:baseline>¶
····· </cv:recommended_baseline>¶
····· <cv:recommended_baseline>¶
·········· <cv:component>¶
·············· <cv:id>9</cv:id>¶
·············· <cv:name>wheels</cv:name>¶
·········· </cv:component>¶
·········· <cv:baseline>¶
·············· <cv:id/>¶
·············· <cv:name/>¶
·········· </cv:baseline>¶
····· </cv:recommended_baseline>¶
··· </cv:recommended_baselines>¶
</cv:response>¶
```

Note that if no recommended baseline has been set and the Subversion HEAD revision is used, getting the recommended baseline will return no value for the cv:name and cv:id elements.

2.6 Get Baselines

Description

HTTP Request Type	GET
URL	/api/project/<project_id>/baseline/

Get a list of baselines for a particular project.

Arguments

token	Login token

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:baselines>¶
····· <cv:baseline>¶
·········· <cv:id>3</cv:id>¶
·········· <cv:name>r4.3_iteration1</cv:name>¶
·········· <cv:status>baseline_untested</cv:status>¶
····· </cv:baseline>¶
····· <cv:baseline>¶
·········· <cv:id>9</cv:id>¶
·········· <cv:name>r4.5_iteration2</cv:name>¶
·········· <cv:status>baseline_good</cv:status>¶
```

```
····· </cv:baseline>¶
··· </cv:baselines>¶
</cv:response>¶
```

2.7 Get Activities

Description

HTTP Request Type	GET
URL	/api/baseline/<baseline_id>/activity/

Get a list of activity names for a baseline or a range of baselines for a particular project. This expresses the activities integrated for a particular baseline or a range of baselines. It allows you to identify the changes compared to previous baselines.

Arguments

token	Login token
baseline_to_id	(Optional) If baseline_to_id is provided, all activities created for the range of baselines between baseline_id and baseline_to_id will be returned.

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:activities>¶
····· <cv:activity>¶
····· <cv:id>1</cv:id>¶
····· <cv:name>add cd radio</cv:name>¶
····· </cv:activity>¶
····· <cv:activity>¶
····· <cv:id>3</cv:id>¶
····· <cv:name>bore out the heads</cv:name>¶
····· </cv:activity>¶
··· </cv:activities>¶
</cv:response>¶
```

2.8 Get Subversion URLs

Description

HTTP Request Type	GET
URL	/api/project/<project_id>/url/
URL	/api/project/<project_id>/component/<component_id>/url/
URL	/api/component/<component_id>/url/

Get Subversion URLs for various elements. Allows you get URLs for the project integration branch, component-related URLs etc.

Arguments

token	Login token

The information desired is indicated through the information provided through the URL. If a project_id is provided, the project-integration-branch URLs for all components in the project will be returned. If a component_id is provided, the root URL for the component is returned. If both project_id and component_id are provided, the component integration URL for the specified component in the specified project is returned.

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
· <cv:status cv:success="true">ok</cv:status>¶
· <cv:urls>¶
··· <cv:url>http://svn.mycompany.com/svn/ucm4svn/Lion_Cage_1/branches/¶
projects/clearvision_zoo_1</cv:url>¶
···
<cv:url>http://svn.mycompany.com/svn/ucm4svn/Monkey_Cage_2/branches/¶
projects/clearvision_zoo_1</cv:url>¶
···
<cv:url>http://svn.mycompany.com/svn/ucm4svn/Penguin_Pool_3/branches/¶
projects/clearvision_zoo_1</cv:url>¶
· </cv:urls>¶
</cv:response>¶
```

2.9 Get Activity Changeset

Description

HTTP Request Type	GET
URL	/api/activity/<activity_id>/changeset/

Get the Subversion changeset for an activity. The changeset shows the changes performed in Subversion for a particular activity.

The changeset itself is returned in the XML format provided by Subversion on `svn diff --xml`. See Subversion documentation for further details.

Arguments

token	Login token

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
```

```
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
· <cv:status cv:success="true">ok</cv:status>¶
· <cv:changeset>¶
··· <cv:component>¶
····· <cv:id>3</cv:id>¶
····· <cv:name>Monkey_Cage</cv:name>¶
····· <cv:svn_changeset>¶
······· <diff>¶
········· <paths>¶
········· <path item="modified" kind="file" props="none">¶
http://localhost/svn/ucm4svn-testing/Monkey_Cage_1/branches/activities/¶
clearvisionzoo_1/Add_Monkey_Swing_1/monkey_swing.txt¶
········· </path>¶
········· <path item="added" kind="file" props="none">¶
http://localhost/svn/ucm4svn-testing/Monkey_Cage_1/branches/activities/¶
clearvisionzoo_1/Add_Monkey_Swing_1/monkey_swing_matt.txt¶
········· </path>¶
········· <path item="added" kind="file" props="none">¶
http://localhost/svn/ucm4svn-testing/Monkey_Cage_1/branches/activities/¶
clearvisionzoo_1/Add_Monkey_Swing_1/monkey_swing_tushar.txt¶
········· </path>¶
········· </paths>¶
······· </diff>¶
··· </cv:component>¶
··· <cv:component>¶
····· <cv:id>9</cv:id>¶
····· <cv:name>Penguin_Pool</cv:name>¶
····· <cv:svn_changeset>¶
······· <diff>¶
········· <paths>¶
········· <path item="added" kind="file" props="none">¶
http://localhost/svn/ucm4svn-testing/Penguin_Pool_2/branches/activities/¶
clearvisionzoo_1/Add_Monkey_Swing_1/monkey_swing_for_penguins.txt¶
········· </path>¶
········· </paths>¶
······· </diff>¶
····· </cv:svn_changeset>¶
··· </cv:component>¶
· </cv:changeset>¶
</cv:response>¶
```

2.10 Get Checkout Commands

Description

HTTP Request Type	GET
URL	/api/baseline/<baseline_id>/checkout/
URL	/api/project/<project_id>/checkout/

Get Subversion checkout/export commands that can be used to create a workspace or file-system copy of the

requested entity. This allows you to export baselines to a file-system location or checkout the project integration branch for a project.

The target location to export or check out to is specified through the `workspace_path` option.

Arguments

token	Login token
workspace_path	The target location to check out/export to.

This function returns a Subversion commands to checkout/export the components that are part of a particular project or form a particular baseline. For projects, commands are provided to checkout the project integration branch. For baselines, commands are returned that export the baseline to a directory structure. Baselines are not checked out as they should never be modified in Subversion.

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">ok</cv:status>¶
··· <cv:checkout_commands>¶
····· <cv:checkout_command>¶
svn co "http://localhost/svn/ucm4svn-testing/Monkey_Cage_1/branches/¶
projects/clearvisionzoo_1" "ClearvisionZoo_1/Monkey_Cage"¶
····· </cv:checkout_command>¶
····· <cv:checkout_command>¶
svn co "http://localhost/svn/ucm4svn-testing/Penguin_Pool_2/branches/¶
projects/clearvisionzoo_1" "ClearvisionZoo_1/Penguin_Pool"¶
····· </cv:checkout_command>¶
··· </cv:checkout_commands>¶
</cv:response>¶
```

2.11 Create Baseline

Description

HTTP Request Type	POST
URL	/api/project/<project_id>/baseline/

Create a new baseline from the current project integration branch for the project specified.

Arguments

token	Login token
baseline_name	Name for the new baseline.
baseline_status	(Optional) Indicate baseline status, which is one of "baseline_good", "baseline_bad", "baseline_untested", "baseline_failed", "baseline_obsolete".

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">Baseline created
successfully</cv:status>¶
</cv:response>¶
```

2.12 Recommend Baselines

Description

HTTP Request Type	PUT
URL	/api/project/<project_id>/
URL	/api/project/<project_id>/component/<component_id>/

Recommend a baseline for all components in a project or an individual component in a project.

Note that recommending a baseline is a modification of an existing project, thus following the principles of RESTful web services it requires a PUT request.

Arguments

token	Login token
baseline_id	The baseline to be used as the recommended baseline for all components.

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">Recommended baseline set
successfully</cv:status>¶
</cv:response>¶
```

2.13 Change Baseline Status

Description

HTTP Request Type	PUT
URL	/api/baseline/<baseline_id>/

Update the status of a baseline.

Arguments

token	Login token
baseline_status	The new baselines status, one of "baseline_good", "baseline_bad", "baseline_untested",

	"baseline_failed", "baseline_obsolete".

Response

```
<?xml version="1.0" encoding="UTF-8"?>¶
<cv:response
xmlns:cv="http://www.clearvision-cm.com/ucm4svn/name_space/cv">¶
··· <cv:status cv:success="true">Baseline status updated</cv:status>¶
</cv:response>¶
```