Mobicents xcap-diff Protocol User Guide

by Eduardo Martins, Bartosz Baranowski, and Alexandre Mendonça

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts* [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press **Enter** to execute the command.
>
> Press **Ctrl**+**Alt**+**F1** to switch to the first virtual terminal. Press **Ctrl**+**Alt**+**F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type `ssh` *`username@domain.name`* at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type `ssh john@example.com`.
>
> The `mount -o remount` *`file-system`* command remounts the named file system. For example, to remount the `/home` file system, the command is `mount -o remount /home`.
>
> To see the version of a currently installed package, use the `rpm -q` *`package`* command. It will return a result as follows: *`package-version-release`*.

Note the words in bold italics above username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as

a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

2. Provide feedback to the authors!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in the the *Issue Tracker* [http://code.google.com/p/mobicents/issues/list], against the product **Mobicents xcap-diff Protocol**, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: XCAPDiff_Protocols_User_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Introduction to xcap-diff Protocol

`XCAP` stands for Extensible Markup Language (`XML`) Configuration Access Protocol. It is used to read, write and modify documents published on `XCAP` enabled servers. Such servers maintain said documents along with list of clients subscribed to the changes. Each change to document on server requires server to notify registered(subscribed) clients. In simplest scenario, client, after it has been notified, rereads whole document from server with `XCAP` call. However this is inefficient since documents may be big, client may be interested only in part of the document - in particular cases change to that part may not happen. Moreover, frequent pulls on `XCAP` resources, increase dramatically amount and size of network traffic.

To counter this problem `xcap-diff` has been created. It is augmented version of `XML Diff` . In general , diff function allows server to generate difference patch and send to clients only changed content, and only in cases when client is interested in change. This approach reduces not only traffic content size, but also number of messages exchanged between server and registered clients.

The `xcap-diff` generates minimal difference patch between two documents or their elements based on part to which client subscribed. Generated patch can be applied to not changed document(ie. at client side) in order to have identical document as one from which patch has been generated.

Design Overview

The `xcap-diff` is defined in *http://tools.ietf.org/html/rfc5875* . However to understand it fully you need to have understanding of following:

XML

XPath
: `XPath` expression are used to navigate through `XML` document and identify specific nodes in it. `XPath` is defined in this *http://www.w3.org/TR/xpath20/* document.

XCAP
: Minimal knowledge how `XCAP` interaction looks like is required. This protocol is defined in *http://tools.ietf.org/html/rfc4825*

XML Diff/XML Patch
: `XCAP Diff` is an extension to `XML Diff` . Strong knowledge of the latter is required. This protocol is defined in *http://tools.ietf.org/html/rfc5261*

SIP Notification framework
: *http://tools.ietf.org/html/rfc3265*

XCAP Diff Notification framework
: *http://tools.ietf.org/html/rfc5875*

The `xcap-diff` has been divided into following components:

XML Patch instructions builder
: component which is capable of creating patching instructions consistent with `rfc5261` .

Attribute Patch instruction builder
: component which is capable of creating `xcap-diff` attribute patching instructions

Element Patch instruction builder
: component which is capable of creating `xcap-diff` element patching instructions

Document Patch instruction builder
: component which is capable of creating `xcap-diff` document wide patch instructions(in particular aggregate instructions generated by *Section 2.1, “XML Patch Instruction Builder”*)

`XCAP` Patch builder
: simple component which builds well formed document from patching operations.

`XCAP` Patch applier
: simple component which is capable of applying patching instructions to document.

The `API` definition for all above components make use of Java generics. Reason for this is to make this library agnostic to specific implementation of `XML` manipulation library.

2.1. XML Patch Instruction Builder

XML Patch instruction builder is defined as follows:

```
public interface XmlPatchOperationsBuilder<D,P, E, N> {


 publicstaticfinalStringXML_PATCH_OPS_NAMESPACE="urn:ietf:params:xml:schema:patch-
ops";

  public enum Pos {
    prepend, before, after
  }

  public enum Ws {
    before, after, both
  }

  public P addAttribute(String sel, String attrName, String attrValue,
      Map<String, String> namespaceBindings) throws BuildPatchException;

  public P addElement(String sel, E element,
      Map<String, String> namespaceBindings) throws BuildPatchException;

  public P addNode(String sel, Pos pos, N node,
      Map<String, String> namespaceBindings) throws BuildPatchException;

  public P addPrefixNamespaceDeclaration(String sel, String namespacePrefix,
      String namespaceValue, Map<String, String> namespaceBindings)
      throws BuildPatchException;

  public P replaceAttribute(String sel, String attributeValue,
      Map<String, String> namespaceBindings) throws BuildPatchException;

  public P replaceElement(String sel, E element,
      Map<String, String> namespaceBindings) throws BuildPatchException;

  public P replaceNode(String sel, N node,
      Map<String, String> namespaceBindings) throws BuildPatchException;
    public P replacePrefixNamespaceDeclaration(String sel,
            String namespaceValue, Map<String, String> namespaceBindings)
            throws BuildPatchException;

    public P removeAttribute(String sel, Map<String, String> namespaceBindings)
            throws BuildPatchException;

    public P removeElement(String sel, Ws ws,
            Map<String, String> namespaceBindings) throws BuildPatchException;

    public P removeNode(String sel, Map<String, String> namespaceBindings)
            throws BuildPatchException;

    public P removePrefixNamespaceDeclaration(String sel,
            Map<String, String> namespaceBindings) throws BuildPatchException;

    public P[] buildPatchInstructions(D originalDocument, D patchedDocument)
            throws BuildPatchException;
}
```

Above code uses following generics:

D
: the document type, defines what is the concrete type of XML documents used

P
: the patching instruction type

E
: the element type

N
: the node type

public P addAttribute(String sel, String attrName, String attrValue, Map<String, String> namespaceBindings) throws BuildPatchException;
: Creates patch operation to add new attribute into document.

public P addElement(String sel, E element, Map<String, String> namespaceBindings) throws BuildPatchException;
: Creates patch operation to add new element into document.

public P addNode(String sel, N node, Map<String, String> namespaceBindings) throws BuildPatchException;
: Creates patch operation to add new node into document.

public P addPrefixNamespaceDeclaration(String sel, String namespacePrefix, String namespaceValue, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to insert new namespace into document.

public P replaceAttribute(String sel, String attributeValue, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to replace existing attribute.

public P replaceElement(String sel, E element, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to replace exisiting element in document.

public P replaceNode(String sel, N node, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to replace existing node in document.

public P replacePrefixNamespaceDeclaration(String sel, String namespaceValue, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to replace exisiting namespace.

public P removeAttribute(String sel, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to remove exisiting attribute from document.

public P removeElement(String sel, Ws ws, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to remove existing element from document.

public P removeNode(String sel, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to remove existing node from document.

public P removePrefixNamespaceDeclaration(String sel, Map<String, String> namespaceBindings) throws BuildPatchException;

Creates patch operation to remove existing namespace from document.

public P[] buildPatchInstructions(D originalDocument, D patchedDocument) throws BuildPatchException;

Compares two documents and creates set of patching operations that need to be applied to first document to be equal to patched one.

2.2. XCAP Attribute Patch Builder

XCAP Attribute Patch instruction builder is defined as follows

```
public interface AttributePatchComponentBuilder<P> {
    public P buildPatchComponent(String sel, String attributeValue,
            Map<String, String> namespaceBindings) throws BuildPatchException;

    public P buildPatchComponent(String sel,
            Map<String, String> namespaceBindings) throws BuildPatchException;

}
```

Above code uses following generics:

P
: the patching instruction type

public P buildPatchComponent(String sel, String attributeValue, Map<String, String> namespaceBindings) throws BuildPatchException;
: Creates patch operation which indicates value of certain attribute in document.

public P buildPatchComponent(String sel, Map<String, String> namespaceBindings) throws BuildPatchException;
: Creates patch operation to indicate certain attribute does not exist in document.

2.3. XCAP Element Patch Builder

XCAP Element Patch instruction builder is defined in very similar manner as Attribute one:

```
public interface ElementPatchComponentBuilder<C, E> {

    public P buildPatchComponent(String sel, E element,
            Map<String, String> namespaceBindings) throws BuildPatchException;


    public P buildPatchComponent(String sel, boolean exists,
            Map<String, String> namespaceBindings) throws BuildPatchException;

}
```

Above code uses following generics:

P
: the patching instruction type

public P buildPatchComponent(String sel, String attributeValue, Map<String, String> namespaceBindings) throws BuildPatchException;
: Creates patch operation which indicates value of certain element in document.

public P buildPatchComponent(String sel, boolean exists, Map<String, String> namespaceBindings) throws BuildPatchException;
: Creates patch operation to indicate if certain element exists in document.

2.4. XCAP Document Patch Builder

XCAP Document Patch instruction builder is defined as follows:

```
public interface DocumentPatchComponentBuilder<P, D, E, N> {

   public P buildPatchComponent(String sel, String previousETag,
         String newETag, C[] patchingInstructions)
         throws BuildPatchException;

   public P getBodyNotChangedPatchingInstruction() throws BuildPatchException;

   public XmlPatchOperationsBuilder<D,P, E, N> getXmlPatchOperationsBuilder();

}
```

Above code uses following generics:

P
: the document patching instruction type

C
: the patching instructions used as elements which build component created

E
: the element type, defines what is the concrete type of XML elements used

N
: the xml patch ops node type, defines what is the concrete type of XML patch ops node params used by {@link XmlPatchOperationsBuilder}

public P buildPatchComponent(String sel, String previousETag, String newETag, C[] patchingInstructions) throws BuildPatchException;

Creates patch instruction for changes in particular document.

public P getBodyNotChangedPatchingInstruction() throws BuildPatchException;

Creates patch instruction indicating that document body has not changed. It is used to indicate tag changes.

public XmlPatchOperationsBuilder<D,P, E, N> getXmlPatchOperationsBuilder();

Returns `XML` Patch instructions builder. Instance of this interface should be used to create instructions to fed to above methods.

2.5. XCAP Patch Builder

`XCAP` Patch instruction builder is defined as follows:

```
public interface XcapDiffPatchBuilder<P, C, D, E, N> {

    public P buildPatch(String xcapRoot, C[] components)
            throws BuildPatchException;

    public AttributePatchComponentBuilder<C> getAttributePatchComponentBuilder();

    public ElementPatchComponentBuilder<C, E> getElementPatchComponentBuilder();

    public DocumentPatchComponentBuilder<C, D, E, N> getDocumentPatchComponentBuilder();

}
```

Above code uses following generics:

P

the patch type, defines what is the concrete type of finalized XCAP DIFF patch

C

the component type, defines what is the concrete type of each patch component, to be aggregated in a XCAP DIFF patch

D

the document type, defines what is the concrete type of XML documents used

E

the element type, defines what is the concrete type of XML elements used

N
: the xml patch ops node type, defines what is the concrete type of XML patch ops node params used by {@link XmlPatchOperationsBuilder}

public P buildPatch(String xcapRoot, C[] components) throws BuildPatchException;
: Creates patch instruction from component passes. Components passed to this method are genereated by interfaces retrieved by getter methods explaioned below.

public AttributePatchComponentBuilder<C> getAttributePatchComponentBuilder();
: Retrieves Attribute Patch builder: *Section 2.2, “XCAP Attribute Patch Builder”*

public ElementPatchComponentBuilder<C, E> getElementPatchComponentBuilder();
: Retrieves Element Patch builder: *Section 2.3, “XCAP Element Patch Builder”*

public DocumentPatchComponentBuilder<C, D, E, N> getDocumentPatchComponentBuilder();
: Retrieves Document Patch builder: *Section 2.4, “XCAP Document Patch Builder”*

2.6. XCAP Patch Applier

`XCAP` Patch instruction applier is defined as follows:

```
public interface XcapDiffPatchApplier<P, D> {

    public void applyPatch(P patch, D document) throws ApplyPatchException;

}
```

Above code uses following generics:

P
: the patch type, defines what is the concrete type of finalized XCAP DIFF patch

D
: the document type, defines what is the concrete type of XML documents used

Applier has single method which applies patch generated to local copy of document to update its content.

2.7. XCAP Diff Factory

Last element defined by xcap-diff library is `XCAP Diff` factory. Simple component which takes care of initialization of all above elements. It is defined as follows:

```
public interface XcapDiffFactory<P, C, D, E, N> {

   public static final String XCAP_DIFF_NAMESPACE_URI = "urn:ietf:params:xml:ns:xcap-diff";

    public XcapDiffPatchApplier<P, D> getPatchApplier();

    public XcapDiffPatchBuilder<P, C, D, E, N> getPatchBuilder();
}
```

Above code uses following generics:

P
: the patch type, defines what is the concrete type of finalized XCAP DIFF patch

C
: the component type, defines what is the concrete type of each patch component, to be aggregated in a XCAP DIFF patch

D
: the document type, defines what is the concrete type of XML documents used

E
: the element type, defines what is the concrete type of XML elements used

N
: the xml patch ops node type, defines what is the concrete type of XML patch ops node params used by {@link XmlPatchOperationsBuilder}

Setup

3.1. Software Prerequisites

There are no specific software requirements that need to be met, other than required by specific implementation of `API` . Currently following implementation exist:

DOM
: It requires following libraries:

  - Xerces

3.2. Mobicents Source Code

3.2.1. Release Source Code Building

1. **Downloading the source code**

   Important

   Subversion is used to manage its source code. Instructions for using Subversion, including install, can be found at *http://svnbook.red-bean.com*

   Use SVN to checkout a specific release source, the base URL is http://mobicents.googlecode.com/svn/tags/protocols/xcap-diff, then add the specific release version, lets consider 1.0.0.BETA1.

```
[usr]$ svn co http://mobicents.googlecode.com/svn/tags/protocols/xcap-diff/xcap-diff-1.0.0.BETA1 xcap-diff-1.0.0.BETA1
```

2. **Building the source code**

   Important

   Maven 2.0.9 (or higher) is used to build the release. Instructions for using Maven2, including install, can be found at *http://maven.apache.org*

   Use Maven to build the binaries.

```
[usr]$ cd xcap-diff-1.0.0.BETA1
[usr]$ mvn install
```

Once the process finishes you should have the `binary` jar files in the `target` directory of `module` .

3.2.2. Development Trunk Source Building

Similar process as for *Section 3.2.1, “Release Source Code Building”*, the only change is the SVN source code URL, which is http://mobicents.googlecode.com/svn/trunk/protocols/xcap-diff.

3.3. Configuration

There is no configuration required to use xcap-diff library.

Examples

Following examples are based on current, default DOM implementation of xcap-diff library. However they are valid for any implementation honoring javadoc contracts.

4.1. Attribute Replace Patch

Example code assumes following:

- Client subscribed to document/element
- Attribute has been replaced in document or in child element to which client subscribed

```
import org.mobicents.protocols.xcap.*; //imports api and default, dom implementation
import org.w3c.dom.*; //import dom  stuff

DOMXcapDiffFactory xcapDiffFactory = new DOMXcapDiffFactory();
DOMXcapDiffPatchBuilder xcapDiffPatchBuilder = xcapDiffFactory.getPatchBuilder();
DOMDocumentPatchComponentBuilder documentPatchComponentBuilder =
   xcapDiffPatchBuilder.getDocumentPatchComponentBuilder();
DOMXmlPatchOperationsBuilder xmlPatchOperationsBuilder =
   documentPatchComponentBuilder.getXmlPatchOperationsBuilder();
//Note, no namespaces
String xcapRoot = "http://localhost:8080/default"; //root of server
String documentSelector = "tests/users/sip:joe@example.com/index";
String attributeSelector = "house/room[id='main']/switch/@on"; //house status document?
String attributeNewValue = "true";
String oldETag = "3416134yyDFGA$v33@!";
String newETag = "haha";

Element attributeElement = xmlPatchOperationsBuilder
   .replaceAttribute(attributeSelector,attributeNewValue,null); //no namespaces

Element documentElement = documentPatchComponentBuilder
   .buildPatchComponent(documentSelector,oldETag,newETag,
      new Element[]{attributeElement});

Document xcapDiffDocument = xcapDiffPatchBuilder
   .buildPatch(xcapRoot,new Element[]{documentElement}); //xcap diff patch
```

4.2. Attribute Add Patch

Example code assumes following:

- Client subscribed to document/element
- Attribute has been added in document or in child element to which client subscribed

```
import org.mobicents.protocols.xcap.*; //imports api and default, dom implementation
import org.w3c.dom.*; //import dom  stuff

DOMXcapDiffFactory xcapDiffFactory = new DOMXcapDiffFactory();
DOMXcapDiffPatchBuilder xcapDiffPatchBuilder = xcapDiffFactory.getPatchBuilder();
DOMDocumentPatchComponentBuilder documentPatchComponentBuilder =
    xcapDiffPatchBuilder.getDocumentPatchComponentBuilder();
DOMXmlPatchOperationsBuilder xmlPatchOperationsBuilder =
    documentPatchComponentBuilder.getXmlPatchOperationsBuilder();
//Note, no namespaces
String xcapRoot = "http://localhost:8080/default"; //root of server
String documentSelector = "tests/users/sip:joe@example.com/index";
String nodeSelector = "house/room[id='main']/switch"; //house status document?
String attributeName = "serial";
String attributeNewValue = "GID-FH56-6235-ZXOP";
String oldETag = "3416134yyDFGA$v33@!";
String newETag = "haha";

Element attributeElement = xmlPatchOperationsBuilder
    .addAttribute(nodeSelector,attributeName,attributeNewValue,null); //no namespaces

Element documentElement = documentPatchComponentBuilder
    .buildPatchComponent(documentSelector,oldETag,newETag,
        new Element[]{attributeElement});

Document xcapDiffDocument = xcapDiffPatchBuilder
    .buildPatch(xcapRoot,new Element[]{documentElement}); //xcap diff patch
```

4.3. Attribute Add Patch

Example code assumes following:

- Client subscribed to attribute
- Attribute has been added

```
import org.mobicents.protocols.xcap.*; //imports api and default, dom implementation
import org.w3c.dom.*; //import dom  stuff

DOMXcapDiffFactory xcapDiffFactory = new DOMXcapDiffFactory();
DOMXcapDiffPatchBuilder xcapDiffPatchBuilder = xcapDiffFactory.getPatchBuilder();

DOMAttributePatchComponentBuilder attributePatchComponentBuilder =
    xcapDiffPatchBuilder.getAttributePatchComponentBuilder();
//Note, no namespaces
String xcapRoot = "http://localhost:8080/default"; //root of server
String documentSelector = "tests/users/sip:joe@example.com/index";
String attributeSelector = "house/room[id='main']/switch/@serial"; //house status document?

String attributeNewValue = "GID-FH56-6235-ZXOP";

Element attributeElement = attributePatchComponentBuilder
    .buildPatchComponent(attributeSelector,attributeNewValue,null); //no namespaces


Document xcapDiffDocument = xcapDiffPatchBuilder
    .buildPatch(xcapRoot,new Element[]{attributeElement}); //xcap diff patch
```

Note

Note difference between this code above and *Section 4.2, “Attribute Add Patch”* Similar rules apply to element changes.

4.4. Element Replace Patch

Example code assumes following:

- Client subscribed to document/element
- Attribute has been replaced in document or in child element to which client subscribed

```
import org.mobicents.protocols.xcap.*; //imports api and default, dom implementation
import org.w3c.dom.*; //import dom  stuff

DOMXcapDiffFactory xcapDiffFactory = new DOMXcapDiffFactory();
DOMXcapDiffPatchBuilder xcapDiffPatchBuilder = xcapDiffFactory.getPatchBuilder();
DOMDocumentPatchComponentBuilder documentPatchComponentBuilder =
    xcapDiffPatchBuilder.getDocumentPatchComponentBuilder();
DOMXmlPatchOperationsBuilder xmlPatchOperationsBuilder =
    documentPatchComponentBuilder.getXmlPatchOperationsBuilder();
//Note, no namespaces
String xcapRoot = "http://localhost:8080/default"; //root of server
String documentSelector = "tests/users/sip:joe@example.com/index";
String nodeSelector = "house/room[id='main']/switch"; //house status document?
Element elementNewValue = .....;
String oldETag = "3416134yyDFGA$v33@!";
String newETag = "haha";

Element element = xmlPatchOperationsBuilder
    .replaceElement(nodeSelector,elementNewValue,null); //no namespaces

Element documentElement = documentPatchComponentBuilder
    .buildPatchComponent(documentSelector,oldETag,newETag,
        new Element[]{element});

Document xcapDiffDocument = xcapDiffPatchBuilder
    .buildPatch(xcapRoot,new Element[]{documentElement}); //xcap diff patch
```

4.5. XPath and Namespaces

The `XPath` expressions may include namespace prefixes to identify correct resource in document. In such cases `XCAP Diff` requires namespace declaration its supposed to include in patch. This is required so patch consumers can properly resolve prefixes used in `XPath`, in order to patch locate target resource in document.

Below is example of patch generation with namespaces in `XPath`. Note that is similar to ???

```


        import org.mobicents.protocols.xcap.*; //imports api and default, dom implementation
        import org.w3c.dom.*; //import dom  stuff

        DOMXcapDiffFactory xcapDiffFactory = new DOMXcapDiffFactory();
        DOMXcapDiffPatchBuilder xcapDiffPatchBuilder = xcapDiffFactory.getPatchBuilder();
        DOMDocumentPatchComponentBuilder documentPatchComponentBuilder =
           xcapDiffPatchBuilder.getDocumentPatchComponentBuilder();
        DOMXmlPatchOperationsBuilder xmlPatchOperationsBuilder =
           documentPatchComponentBuilder.getXmlPatchOperationsBuilder();
        //Note, no namespaces
        String xcapRoot = "http://localhost:8080/default"; //root of server
        String documentSelector = "tests/users/sip:joe@example.com/index";
                String attributeSelector = "p:house/room[id='main']/e:switch/@on"; //house status
document?
        String attributeNewValue = "true";
        String oldETag = "3416134yyDFGA$v33@!";
        String newETag = "haha";
        Map<String,String> nameSpaces = new HashMap<String,String>();
        nameSpaces.put("e","http://engineer.org/grid/electrical");
        nameSpaces.put("p","http://engineer.org/ownership/private");


        Element attributeElement = xmlPatchOperationsBuilder
           .replaceAttribute(attributeSelector,attributeNewValue,nameSpaces); //no namespaces

        Element documentElement = documentPatchComponentBuilder
           .buildPatchComponent(documentSelector,oldETag,newETag,
              new Element[]{attributeElement});

        Document xcapDiffDocument = xcapDiffPatchBuilder
           .buildPatch(xcapRoot,new Element[]{documentElement}); //xcap diff patch
```

Appendix A. Revision History

Revision History

Revision 1.0	Fri June 30 2011	BartoszBaranowski

Creation of the Mobicents xcap-diff Protocol User Guide.

Index