

**pharos**  
communications limited

# Mediator Web Services - Accessing with PHP

Pharos Mediator Web Services - PHP Access v1.0.docx

Jeremy Blythe (Steve Robinson)  
08/07/2008

## Revision History

### Pharos Mediator Web Services - PHP Access v1.0.docx

<b>Version</b>	<b>Date</b>	<b>Comments</b>	<b>Author</b>
1.0	8th July	Word/PDF version of the original web article	Jeremy Blythe (Steve Robinson)

---

## Contents

---

<b>1</b>	<b>Pharos Mediator Web Services</b>	<b>4</b>
1.1	Introducing - Accessing with PHP	4

---

# 1 Pharos Mediator Web Services

---

## 1.1 Introducing - Accessing with PHP

This article introduces the basic concepts of the Pharos Mediator XML API and web service. An example written in PHP is used to demonstrate a single method call. It is useful for anyone interested in learning the basics of the API and particularly those wishing to integrate Mediator with an existing Internet or Intranet site.

### Command structure

At the heart of the XML API is the PharosCs document type. Mediator can receive and process PharosCs documents from many sources but the XML document is the same type whether it's read from a file system or received over HTTP. The document contains a list of commands with their input parameters. Mediator processes each command and returns a document containing the list of commands with their corresponding output data.

Example PharosCs document sent to Mediator for processing:

```
<PharosCs>
  <CommandList>
    <Command subsystem="material" method="get" reference="get1">
      <ParameterList>
        <Parameter name="material">
          <Value>
            <Material>
              <MatId>TEST</MatId>
            </Material>
          </Value>
        </Parameter>
      </ParameterList>
    </Command>
  </CommandList>
</PharosCs>
```

Focussing on the Command element we can see that a method "get" is to be called on subsystem "material". There is a single parameter named "material" to which we have given a Material object with MatId "TEST". Subsystems are used to categorise sets of methods either by function or by object. So the functional subsystem "library" contains methods "getLocation", "getDespatch" and "moveMedia" where as the "material" object subsystem has methods "get", "save", "delete" etc.

The reply from Mediator (some elements have been removed for brevity):

```

<PharosCs>
  <CommandList>
    <Command method="get" reference="get1" subsystem="material">
      <Output>
        <Material>
          <TrackTypeLink>
            <TrackTypeName>Video</TrackTypeName>
            <StateName>Preview required</StateName>
          </TrackTypeLink>
          <TrackTypeLink>
            <TrackTypeName>M&amp;E L</TrackTypeName>
            <StateName>Preview required</StateName>
          </TrackTypeLink>
          <TrackTypeLink>
            <TrackTypeName>M&amp;E R</TrackTypeName>
            <StateName>Preview required</StateName>
          </TrackTypeLink>
          <TrackTypeLink>
            <TrackTypeName>hun L</TrackTypeName>
            <StateName>Preview required</StateName>
          </TrackTypeLink>
          <TrackTypeLink>
            <TrackTypeName>hun R</TrackTypeName>
            <StateName>Preview required</StateName>
          </TrackTypeLink>
          <MaterialType>Branding</MaterialType>
          <Title>The title for the test material</Title>
          <Trim>00:00:00:00</Trim>
          <AspectRatio>4:3</AspectRatio>
          <Duration>00:00:38:00</Duration>
          <MatId>TEST</MatId>
        </Material>
      </Output>
    </Command>
  </CommandList>
</PharosCs>

```

Here we can see the original reference value has been returned in the reply so if needed our application can match a request to a response. If a command causes an error it is returned as a Command Exception.

Below is the output if we try to call the method "banana" which doesn't exist:

```
<PharosCs>
  <CommandList>
    <Command method="banana" reference="get1" subsystem="material">
      <Output>
        <CommandException>
          <Code>methodNotFound</Code>
          <Cause/>
          <Message>The method [banana] could not be found</Message>
        </CommandException>
      </Output>
    </Command>
  </CommandList>
</PharosCs>
```

Client applications should always check for the presence of the "CommandException" element in every reply from Mediator.

### PHP example

PHP is a popular scripting language for both Internet and Intranet sites. This example shows how simple it would be to enhance an existing site with live data from Mediator.

While the PharosCs XML can be de-serialized to objects (as it is inside Mediator itself) this is rarely the best approach for all but the most complex consumers of the service. Instead a solution that works directly with the XML is encouraged and the simplicity of the PharosCs document promotes this. By working directly on the XML, consumers can be loosely tied to the service: they don't need to maintain a complete set of compatible objects. This methodology for consumers of web services is aided by tools now available for many different languages. In PHP the [SimpleXML](#) extension is used:

**"The SimpleXML extension provides a very simple and easily usable toolset to convert XML to an object that can be processed with normal property selectors and array iterators."**

The following single PHP script provides a form to lookup material by material id. The submitted material id is sent to Mediator using the "material" subsystem's "get" method presented earlier. The result from Mediator is first checked for exceptions and then presented back to the user as HTML tables.

Material:

Title	Duration
The title for the test material	00:00:38:00

Track type	Workflow state
Video	Preview required
M&E L	Preview required
M&E R	Preview required
hun L	Preview required
hun R	Preview required

```
<?php
// If this is a POST then process the form data
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Get the material id from the form for lookup
    $lookupMatId = $_POST["matId"];

    // Make a SimpleXML object for the material.get method
    $xmlCmd = new SimpleXMLElement('<?xml version="1.0"?>
<PharosCs>
    <CommandList>
        <Command subsystem="material" method="get" reference="get1">
            <ParameterList>
                <Parameter name="material">
                    <Value>
                        <Material>
                            <MatId></MatId>
                        </Material>
                    </Value>
                </Parameter>
            </ParameterList>
        </Command>
    </CommandList>
</PharosCs>');

    // Plug the material id from the form into the SimpleXML object,
    // this is safer than placing it in the string above:
    // the material id might contain characters that need to be escaped.
    $xmlCmd->CommandList->Command->ParameterList->Parameter->Value->Material->MatId
        = $lookupMatId;

    //continue on next page...
```

```

// Make the HTTP request to Mediator
$options = array('http' =>
    array(
        'method' => 'POST',
        'header' => 'Content-type: text/xml;',
        'content' => $xmlCmd->asXML()
    )
);
$context = stream_context_create($options);
$xmlstr = file_get_contents('http://localhost/mediator/main/ws', false, $context);

// Make a SimpleXML object from the response
$xml = new SimpleXMLElement($xmlstr);

// Check the response for a command exception
if (isset($xml->CommandList->Command->Output->CommandException)) {
    $e = $xml->CommandList->Command->Output->CommandException;
} else {
    // If there wasn't an exception then set the mat variable to the Material
    // element from the XML
    $mat = $xml->CommandList->Command->Output->Material;
}
} else {
    $lookupMatId = "";
}
?>

<html>
<head>
<title>Simple Pharos Web Service example</title>
<style>
    table { text-align: left; width: 500px; margin-bottom: 20px;}
    table td { background-color: #CCC; }
</style>
</head>
<body>
    <?php
        // If there was an exception then print it out at the top of the page
        if (isset($e)) {
            echo "<p>Exception ($e->Code): $e->Message</p>";
        }
    ?>

    <form action="<?php echo $ SERVER['SCRIPT NAME'] ?>" method="POST">
        <label for="matId">Material:</label>
        <input type="text" name="matId" id="matId" value="<?php echo $lookupMatId
?>" />

        <input type="submit" name="submit" value="Lookup" />
    </form>

// continued on next page...

```

```
<?php
// If the material variable has a Material element in it then display it
    if (isset($mat)) {
        ?>

        <table>
            <tr>
                <th>Title</th>
                <th>Duration</th>
            </tr>
            <tr>
                <td><?php echo $mat->Title ?></td>
                <td><?php echo $mat->Duration ?></td>
            </tr>
        </table>
        <table>
            <tr>
                <th>Track type</th>
                <th>Workflow state</th>
            </tr>
            <?php
                // Loop through each TrackTypeLink element and
                // print out the TrackTypeName and StateName
                foreach ($mat->TrackTypeLink as $ttl) {
                    echo "<tr><td> $ttl->TrackTypeName </td><td> $ttl->StateName </td><tr/>";
                }
            ?>
        </table>
    <?php } ?>
</body>
</html>
```

by Jeremy Blythe  
Software Architect  
Pharos Communications