

How to use the Medulla XML-RPC API

This guide will help users utilize Medulla's XML-RPC API: [Medulla's XML-RPC API.php](#)

1: Configuration

Check the following:

- SSL Certificate: Download the certificate from the Medulla server:
http://<medulla_server>/downloads/medulla-ca-chain.cert.pem
- Install the certificate on the server that will be querying the API
 - For Debian-based systems, copy the file to `/usr/local/share/ca-certificates` and change its extension to `crt`, then import the certificate by running the `update-ca-certificates` command
 - For RedHat-based systems, copy the file to `/etc/pki/ca-trust/source/anchors` and change its extension to `crt`, then import the certificate by running the `update-ca-trust extract` command
- If necessary, add the IP address and hostname of the XML-RPC server to `/etc/hosts`

2: Using the API

First authenticate the user and retrieve the session cookie:

```
/**
 * Executes an XML-RPC request.
 *
 * @param string $method The name of the XML-RPC method to call.
 * @param array $params The parameters to pass to the method.
 * @param bool $includeCookie Indicates whether the session cookie should be included in the request.
 * @return array An array containing the HTTP headers and the body of the response.
 * @throws Exception If an error occurs while connecting or sending the request.
 */
function executeRequest($method, $params, $includeCookie = false) {
    $agentInfo = $_SESSION["XMLRPC_agent"];
    $requestXml = xmlrpc_encode_request($method, $params, ['output_type' => 'php', 'verbosity' => 0]);

    // Set the HTTP headers
    $url = "/";
    $httpQuery = "POST " . $url . " HTTP/1.0\r\n";
    $httpQuery .= "User-Agent: MMC web interface\r\n";
    $httpQuery .= "Host: " . $agentInfo["host"] . ":" . $agentInfo["port"] . "\r\n";
```

```

$httpQuery .= "Content-Type: text/xml\r\n";
$httpQuery .= "Content-Length: " . strlen($requestXml) . "\r\n";
// Add the cookie if necessary
if ($includeCookie) {
    $httpQuery .= "Cookie: " . $_SESSION['cookie'] . "\r\n";
}
$httpQuery .= "Authorization: Basic " . base64_encode($agentInfo["login"] . ":" . $agentInfo["password"]);
$httpQuery .= $requestXml;

// Configure the SSL context
// 'allow_self_signed' is set to false to accept only certificates signed by a recognized CA
// 'verify_peer' is set to true to verify the server's SSL certificate
$context = stream_context_create();
$proto = $agentInfo["scheme"] == "https" ? "ssl://" : "";
if ($proto) {
    stream_context_set_option($context, "ssl", "allow_self_signed", false);
    stream_context_set_option($context, "ssl", "verify_peer", true);
}

// Open the connection to the server
$socket = stream_socket_client($proto . $agentInfo["host"] . ":" . $agentInfo["port"], $errno, $errstr);
if (!$socket) {
    throw new Exception("Unable to connect to XML-RPC server: $errstr ($errno)");
}

// Set a 60-second timeout for reading
stream_set_timeout($socket, 60);

if (!fwrite($socket, $httpQuery)) {
    throw new Exception("Unable to send data to XML-RPC server");
}

$responseXml = '';
while (!feof($socket)) {
    $ret = fgets($socket, 128);
    $responseXml .= $ret;
}
fclose($socket);

// Separate the HTTP headers from the response body
list($headers, $body) = explode("\r\n\r\n", $responseXml, 2);
return [$headers, $body];
}

/**
 * Authenticates the user and retrieves the session cookie.
 * @param string $method The name of the XML-RPC method to call.
 * @param array $params The parameters to pass to the method.
 * @return string The session cookie.
 * @throws Exception If an error occurs during authentication.
 */
function authenticateAndGetCookie($method, $params) {
    list($headers, $body) = executeRequest($method, $params);
}

```

```

// Parse the HTTP headers to extract the session cookie
$headers_array = array();
$header_lines = explode("\r\n", $headers);
foreach ($header_lines as $header) {
    $parts = explode(':', $header, 2);
    if (count($parts) == 2) {
        $headers_array[$parts[0]] = $parts[1];
    }
}

// Using the parsed headers
if (isset($headers_array['Set-Cookie'])) {
    $cookie = $headers_array['Set-Cookie'];
    $_SESSION['cookie'] = $cookie;
} else {
    throw new Exception('Authentication failed, no cookie received');
}

return $cookie;
}

```

3: Examples of requests

- Get the list of all packages

```

/**
 * Sends an XML-RPC request and returns the response only if authenticated by cookie.
 * @param string $method The name of the XML-RPC method to call.
 * @param array $params The parameters to pass to the method.
 * @return array The XML-RPC response.
 * @throws Exception If an error occurs while sending the request.
 */
function sendXmlRpcRequest($method, $params) {
    list($headers, $body) = executeRequest($method, $params, true);
    $responseXml = substr($body, strpos($body, '<?xml'));
    $response = xmlrpc_decode($responseXml, 'UTF-8');
    if (is_array($response) && xmlrpc_is_fault($response)) {
        throw new Exception
("XML-RPC fault: {$response['faultString']} ({$response['faultCode']}");
    }
    return $response;
}

$method = "pkgs.get_all_packages";
$params = [
    'root', // login
    false, // sharing_activated
    0, // start
    10, // end

```

```

    [
        'filter' => 'hostname', // example name of package
    ] // ctx
];

try {
    $responseXml = sendXmlRpcRequest($method, $params);

} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}

```

Tips: To get the package UUID: `$responseXml['datas']['uuid'][$key]`

- Get the list of all machines

```

$method = "xmppmaster.get_machines_list";
$params = [
    0, // start
    20, // end
    [
        'filter' => '',
        'field' => 'allchamp',
        'computerpresence' => 'presence', // Specify whether to include machines that are present
        // 'computerpresence' => 'no_presence', // Specify whether to include non-present machines
        'location' => "UUID0", // glpi_id - entity
    ] // ctx
];

try {
    $responseXml = sendXmlRpcRequest($method, $params);
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}

```

- Get a machine's details

```

$method = "glpi.getLastMachineInventoryPart";
$params = [
    $uuid,
    'Summary',
    0, // minbound
    0, // maxbound
    "", // filter
    [
        "hide_win_updates" => false,
        "history_delta" => false
    ], // options
];

```

```

try {
    $responseXml = sendXmlRpcRequest($method, $params);
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}

```

Tips: To get the UUID of the machine: `$responseXml['data']['uuid_inventorymachine'][$key]`. This also corresponds to the GLPI Machine ID prefixed by the UUID

- Deploy a specific package on a target machine

```

$method = "msc.add_command_api";
$pid = "96982fce-hostname_vq918j7wtwzwm610by"; // pid - package_id
$target = "UUID1"; // target - machine_uuid

```

```

$params = [
    $pid, // pid - package_id
    $target, // target - machine_uuid
    array( // params
        "name" => "devdemo-win-1",
        "hostname" => "devdemo-win-1",
        "uuid" => $target,
        "gid" => NULL,
        "from" => "base|computers|msctabs|tablogs",
        "pid" => $pid,
        // "title" => "DEPLOYMENT TITLE",
        "create_directory" => "on",
        "start_script" => "on",
        "clean_on_success" => "on",
        "do_reboot" => "",
        "do_wol" => "",
        "do_inventory" => "on",
        "next_connection_delay" => "60",
        "max_connection_attempt" => "3",
        "maxbw" => "0",
        "deployment_intervals" => "",
        "tab" => "tablaunch",
        "issue_halt_to" => array(),
    ),
    "push", // mode
    NULL, // gid
    array(), // proxy
    0 // cmd_type
];

try {
    $responseXml = sendXmlRpcRequest($method, $params);

    $commandId = $responseXml; // Retrieve the command ID

    $method2 = "xmpmaster.addlogincommand";

```

```

$params2 = [
    'root', // login
    $commandId, // commandid
    '', // grpId
    '', // nb_machine_in_grp
    '', // instructions_nb_machine_for_exec
    '', // instructions_datetime_for_exec
    '', // parameterspackage
    0, // rebootrequired
    0, // shutdownrequired
    0, // bandwidth
    0, // syncthing
    [] // params
];

try {
$responseXml2 = sendXmlRpcRequest($method2, $params2);

} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}

} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
}

```

- Get deployment logs (Audit) from sessionname

```

$method = "xmppmaster.getlinelogsession";
$params = [
    "command63bb5ee8fc834eae89" // sessionname
];

try {
$responseXml = sendXmlRpcRequest($method, $params);
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
}

```

Tips: To get the deployment session name: \$responseXml['tabdeploy']['sessionid'][\$key]

- Get all deployment logs by User and/or target

```

$method = "xmppmaster.get_deploy_by_user_with_interval";
$params = [
    "root", // login_user
    "", // state_deploy
    86400, // intervalsearch
    0, // start_pagination
    "20", // end_pagination
];

```

```
    "spo-win-1", // filt - hostname_machine
    "command"    // typedeploy
];

try {
    $responseXml = sendXmlRpcRequest($method, $params);
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
```

[New boost](#)

Revision #2

Created 2026-04-29 19:03:50 UTC by Adrien Thaisse

Updated 2026-04-29 19:24:26 UTC by Adrien Thaisse