Comment utiliser l'API XML-RPC de Medulla

This will guide the user to using Medulla's XML-RPC API : Medulla's XML-RPC API.php

1: Configuration

Check the following:

- SSL Certificate: Grab the certificate from 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 *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 *update-ca-trust extract* command
- If needed, add the ip and hostname of the XML-RPC server to /etc/hosts

2: Using the API

First authenticate the user and fetch the session cookie:

```
/**
 * Exécute une requête XML-RPC.
 *
 * @param string $method Le nom de la méthode XML-RPC à appeler.
 * @param array $params Les paramètres à passer à la méthode.
 * @param bool $includeCookie Indique si le cookie de session doit être inclus dans la requête.
 * @return array Un tableau contenant les en-têtes HTTP et le corps de la réponse.
 * @throws Exception Si une erreur se produit lors de la connexion ou de l'envoi de la requête.
 */
function executeRequest($method, $params, $includeCookie = false) {
    $agentInfo = $_SESSION["XMLRPC_agent"];
    $requestXml = xmlrpc_encode_request($method, $params, ['output_type' => 'php', 'verbosity' =

    // On définit les en-têtes HTTP
    $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";
    // On ajoute le cookie si nécessaire
    if ($includeCookie) {
        $httpQuery .= "Cookie: " . $_SESSION['cookie'] . "\r\n";
    }
    $httpQuery .= "Authorization: Basic " . base64_encode($agentInfo["login"] . ":" . $agentInfo
    $httpQuery .= $requestXml;

    // Configurer le contexte SSL
    // 'allow_self_signed' est défini sur false pour n'accepter que les certificats signés par u
    // 'verify_peer' est défini sur true pour vérifier le certificat SSL du serveur
    $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);
    }

    // On ouvre la connexion au serveur
    $socket = stream_socket_client($proto . $agentInfo["host"] . ":" . $agentInfo["port"], $errn
    if (!$socket) {
        throw new Exception("Unable to connect to XML-RPC server: $errstr ($errno)");
    }

    // On ajoute un délai de 60 secondes pour la lecture
    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);

    // Séparez les en-têtes HTTP du corps de la réponse
    list($headers, $body) = explode("\r\n\r\n", $responseXml, 2);
    return [$headers, $body];
}

/**
 * Authentifie l'utilisateur et récupère le cookie de session.
 * @param string $method Le nom de la méthode XML-RPC à appeler.
 * @param array $params Les paramètres à passer à la méthode.
 * @return string Le cookie de session.
 * @throws Exception Si une erreur se produit lors de l'authentification.
 */
function authenticateAndGetCookie($method, $params) {
    list($headers, $body) = executeRequest($method, $params);
```

```
    // Parsing des en-têtes HTTP pour extraire le cookie de session
    $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];
        }
    }

    // Utilisation des en-têtes analysés
    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

```
/**
 * Envoie une requête XML-RPC et retourne la réponse uniquement si authentifié par cookie.
 * @param string $method Le nom de la méthode XML-RPC à appeler.
 * @param array $params Les paramètres à passer à la méthode.
 * @return array La réponse XML-RPC.
 * @throws Exception Si une erreur se produit lors de l'envoi de la requête.
 */
function sendXmlRpcRequest($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: $reponseXml['datas']['uuid'][$key]

- Get the list of all machines

```
$method = "xmppmaster.get_machines_list";
$params = [
     0, // start
     20, // end
     [
        'filter' => '',
        'field' => 'allchamp',
        'computerpresence' => 'presence', // Définir si on veut les machines présentes
        // 'computerpresence' => 'no_presence', // Définir si on veut les machines non présentes
        '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: $reponseXml['data']['uuid_inventorymachine'][$key]. This also corresponds to GLPI Machine ID prefixed by UUID

- Deploy a specific package on a target machine

```
$method = "msc.add_command_api";
$pid = "96982fce-hostname_vq918j7wtnwzm610by"; // pid - paquet_id
$target = "UUID1"; // target - uuid_machine

$params = [
    $pid,  // pid - paquet_id
    $target,  // target - uuid_machine
    array(  // params
    "name" => "devdemo-win-1",
    "hostname" => "devdemo-win-1",
    "uuid" => $target,
    "gid" => NULL,
    "from" => "base|computers|msctabs|tablogs",
    "pid" => $pid,
    // "ltitle" => "TITRE DU DEPLOIEMENT",
    "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; // Récupérer l'ID de la commande

        $method2 = "xmppmaster.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.getlinelogssession";
$params = [
    "command63bb5ee8fc834eae89" // sessionname
];

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

*Tips: To get the deployment sessionname: $reponseXml['tabdeploy']*['sessionid'][$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 2026-01-24 10:31:06 UTC by Guillaume Lassarre
Updated 2026-01-24 10:33:46 UTC by Guillaume Lassarre