Cómo utilizar la API XML-RPC de Medulla

Esto guiará al usuario en el uso de la API XML-RPC de Medulla: Medulla's XML-RPC API.php

1: Configuración

Compruebe lo siguiente:

- Certificado SSL: Obtenga el certificado del servidor de Medulla:
  http://<medulla_server>/downloads/medulla-ca-chain.cert.pem
- Instala el certificado en el servidor que va a consultar la API
  - Para sistemas basados en Debian, copie el archivo a*/usr/local/share/ca-certificates*y cambie su extensión a crt; a continuación, importe el certificado ejecutando el comando*update-ca-certificates*
  - Para sistemas basados en RedHat, copie el archivo a*/etc/pki/ca-trust/source/anchors* y cambie su extensión a crt; a continuación, importe el certificado ejecutando el comando*update-ca-trust extract*
- Si es necesario, añada la IP y el nombre de host del servidor XML-RPC a /etc/hosts

2: Uso de la API

Primero autentique al usuario y obtenga la cookie de sesión:

```
/**
 * Ejecuta una solicitud XML-RPC.
 *
 * @param string $method El nombre del método XML-RPC que se va a llamar.
 * @param array $params Los parámetros que se van a pasar al método.
 * @param bool $includeCookie Indica si la cookie de sesión debe incluirse en la solicitud.
 * @return array Un array que contiene los encabezados HTTP y el cuerpo de la respuesta.
 * @throws Exception Si se produce un error al conectarse o al enviar la solicitud.
 */
function executeRequest($method, $params, $includeCookie = false) {
    $agentInfo = $_SESSION["XMLRPC_agent"];
    $requestXml = xmlrpc_encode_request($method, $params, ['output_type' => 'php', 'verbosity' =

    // Definimos los encabezados 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";
    // Se añade la cookie si es necesario
    if ($includeCookie) {
        $httpQuery .= "Cookie: " . $_SESSION['cookie'] . "\r\n";
    }
    $httpQuery .= "Authorization: Basic " . base64_encode($agentInfo["login"] . ":" . $agentInfo
    $httpQuery .= $requestXml;

    // Configurar el contexto SSL
    // 'allow_self_signed' se establece en false para aceptar solo certificados firmados por una
    // 'verify_peer' se establece en true para verificar el certificado SSL del servidor
    $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);
    }

    // Abrimos la conexión con el servidor
    $socket = stream_socket_client($proto . $agentInfo["host"] . ":" . $agentInfo["port"], $errn
    if (!$socket) {
        throw new Exception("No se puede conectar al servidor XML-RPC: $errstr ($errno)");
    }

    // Se añade un tiempo de espera de 60 segundos para la lectura
    stream_set_timeout($socket, 60);

    if (!fwrite($socket, $httpQuery)) {
        throw new Exception("No se pueden enviar datos al servidor XML-RPC");
    }

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

    // Separa los encabezados HTTP del cuerpo de la respuesta
    list($headers, $body) = explode("\r\n\r\n", $responseXml, 2);
    return [$headers, $body];
}

/**
 * Autentifica al usuario y recupera la cookie de sesión.
 * @param string $method El nombre del método XML-RPC que se va a llamar.
 * @param array $params Los parámetros que se van a pasar al método.
 * @return string La cookie de sesión.
 * @throws Exception Si se produce un error durante la autenticación.
 */
function authenticateAndGetCookie($method, $params) {
    list($headers, $body) = executeRequest($method, $params);
```

```
    // Análisis de los encabezados HTTP para extraer la cookie de sesión
    $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];
        }
    }

    // Uso de los encabezados analizados
    if (isset($headers_array['Set-Cookie'])) {
        $cookie = $headers_array['Set-Cookie'];
        $_SESSION['cookie'] = $cookie;
    } else {
        throw new Exception('Autenticación fallida, no se ha recibido ninguna cookie');
    }

    return $cookie;
}
```

3: Ejemplos de solicitudes

- Obtener la lista de todos los paquetes

```
/**
 * Envía una solicitud XML-RPC y devuelve la respuesta solo si se ha autenticado mediante cookie
 * @param string $method El nombre del método XML-RPC que se va a llamar.
 * @param array $params Los parámetros que se van a pasar al método.
 * @return array La respuesta XML-RPC.
 * @throws Exception Si se produce un error al enviar la solicitud.
 */
function sendXmlRpcRequest($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
("Error XML-RPC: {$response['faultString']} ({$response['faultCode']})");
    }
    return $response;
}

    $method = "pkgs.get_all_packages";
    $params = [
        'root', // inicio de sesión
        false,  // sharing_activated
        0,      // inicio
        10,      // fin
```

```
        [
            'filter' => 'hostname', // nombre de ejemplo del paquete
        ] // ctx
    ];

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

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

Consejos: Para obtener el UUID del paquete: $respuestaXml['datas']['uuid'][$key]

- Obtener la lista de todas las máquinas

```
$method = "xmppmaster.get_machines_list";
$params = [
     0, // inicio
     20, // fin
     [
        'filter' => '',
        'field' => 'allchamp',
        'computerpresence' => 'presence', // Definir si se quieren las máquinas presentes
        // 'computerpresence' => 'no_presence', // Definir si se quieren las máquinas no present
        'location' => "UUID0", // glpi_id - entidad
    ] // ctx
];

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

- Obtener los detalles de una máquina

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

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

Consejos: Para obtener el UUID de la máquina: $reponseXml['data']['uuid_inventorymachine'][$key]. Esto también se corresponde con el ID de máquina de GLPI precedido por el UUID

- Implementar un paquete específico en una máquina de destino

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

$params = [
    $pid,  // pid - id_del_paquete
    $target,  // target - uuid_de_la_máquina
    array(  // parámetros
    "name" => "devdemo-win-1",
    "hostname" => "devdemo-win-1",
    "uuid" => $target,
    "gid" => NULL,
    "from" => "base|computers|msctabs|tablogs",
    "pid" => $pid,
    // "title" => "TÍTULO DE LA IMPLEMENTACIÓN",
    "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",  // modo
    NULL,  // gid
    array(),  // proxy
    0  // cmd_type
];

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

        $commandId = $responseXml; // Recuperar el ID del comando
```

```
        $method2 = "xmppmaster.addlogincommand";
        $params2 = [
            'root', // inicio de sesión
            $commandId, // ID del comando
            '', // ID del grupo
            '', // n.º de máquinas en el grupo
            '', // número de máquinas para ejecución
            '', // fecha y hora de ejecución
            '', // paquete de parámetros
            0, // reinicio requerido
            0, // apagado requerido
            0, // ancho de banda
            0, // sincronización
            [] // parámetros
        ];

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

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

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

- Obtener registros de implementación (Audit) de sessionname

```
$method = "xmppmaster.getlinelogssession";
$params = [
    "command63bb5ee8fc834eae89" // nombre de sesión
];

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

Consejos: Para obtener el nombre de sesión de la implementación: $reponseXml['tabdeploy'] ['sessionid'][$key]

- Obtener todos los registros de implementación por usuario y/o destino

```
$method = "xmppmaster.get_deploy_by_user_with_interval";
$params = [
    "root",       // usuario de inicio de sesión
    "",           // estado de implementación
    86400,        // intervalo de búsqueda
```

```
    0,            // inicio de paginación
    "20",         // fin de paginación
    "spo-win-1",  // filtro - nombre_de_host_de_la_máquina
    "command"     // tipo_de_implementación
];

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

Nuevo impulso

Revision #2
Created 2026-04-29 19:04:02 UTC by Adrien Thaissen
Updated 2026-04-29 19:24:32 UTC by Adrien Thaissen