Asynchronous Notifications

Table of Contents

1 General
2 Job Notifications
3 Event Notifications (Notify)
   3.1 Events that trigger an event notification
      3.1.1 TransferOut
      3.1.2 DomainAutoUpdate
      3.1.3 OCVAL (OwnerC validation)
4 Receiving asynchronous System Notifications
   4.1 Configuration in the interface
      4.1.1 Configuration for Emails
      4.1.2 Configuration for Polling
      4.1.3
      4.1.4 Configuration for Push
   4.2 Polling
      4.2.1 Get Notifications
      4.2.2 Examples for JSON, XML and EPP
      4.2.3 Confirm Notifications
      4.2.4 Examples for JSON, XML and EPP
   4.3 Push
      4.3.1 Target addresses
      4.3.2 Source IP of the push notification
      4.3.3 Example PHP script for writing the push message to a textfile
   4.4 Email

General

System notifications are asynchronous notifications that inform you of the status of an order. They are sent with a time delay because other instances, such as the registry, are involved in executing the process.

The system sends two types of asynchronous responses

- Job notifications
- Event notifications

Job Notifications

Job notifications are system responses related to a request sent to the system. For an incoming request, the system immediately sends a system response in real time with the status type `notify`, which reports the start of processing. After processing the job, the system sends an job notification with the final job status, which can be either `success` on successful processing or `error` on failed processing.

Event Notifications (Notify)
Event notifications are notifications that are triggered by external events, such as an outgoing transfer or owner validation. Event notifications are sent to the owner of the object.

**Events that trigger an event notification**

**TransferOut**

- **transfer_start**
  Start of an outgoing domain transfer
- **transfer_autoack**
  Automatic confirmation of an outgoing domain transfer (AutoAck). Only for gTLDs.
- **transfer_autonack**
  Automatic refusal of an outgoing transfer (AutoNack).
- **transferReminder**
  Reminder of an ongoing transfer.
- **domain_away**
  Deletion of a domain from the database, e.g. in the case of domain transfers without the participation of the registrar.

**DomainAutoUpdate**

- **autoupdate_dns_error**
  Successful execution of an automatic update. Only for .de domains. An automatic update is only performed if the connectivity check, which is part of the registration process, has failed.
- **autoupdate_dns_success**
  Successful connectivity check
- **autoupdate_deffered_success**
- **autoupdate_deffered_error**

**OCVAL (OwnerC validation)**

- **ocval_reminder**
  Reminder to deactivate the domain (5 days before).
- **ocval_activation**
  Information about the disconnection of the name servers and the non-accessibility of the domain.
- **ocval_deactivation**
  Information about the reconnection of the nameservers and the re-accessibility of the domain.

**Receiving asynchronous System Notifications**

You can retrieve asynchronous system responses using **Polling** or by **Push** or **Email**.

**Configuration in the interface**

The notifications can be configured in the AutoDNS interface (User Settings / User Profile / API Settings).

**Configuration for Emails**

For **Email** three different notification formats can be defined:

- **Standard**: The content consists of text and XML.
- **XML**: The content consists of pure XML.
- **JSON**: The content is pure JSON.
Configuration for Polling

Three different notification formats can be defined for **Polling**. The set format is used if the message is not retrieved and is sent by the system via email.

- **XML**: The content consists of pure XML.
- **JSON**: The content consists of pure JSON.
- **EPP**: The content consists of pure EPP (will be set automatically as soon as EPP API is activated).

**Polling**

**Polling** is the active retrieval of system notifications by the user. The notifications are available in structured format and are processed according to the FIFO principle.

Get Notifications
**Examples for JSON, XML and EPP**

<table>
<thead>
<tr>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /poll</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
</table>
Confirm Notifications

JSON: A specific system notification can be confirmed via the route PUT /poll/{id}. The ID can be taken from the notification.

XML: A specific system notification can be confirmed with the task 0906. The ID can be taken from the notification.

Hints concerning the successful delivery of a polling notification

Notifications are considered "successfully delivered" if they have been confirmed by the user.

If a notification is not confirmed by the user within 24 hours, it will be delivered to the fallback email (reply_to from the request).

Examples for JSON, XML and EPP

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
</table>
| PUT /poll/{id} | {
  "stid": "20180926-stid",
  "status": {
    "code": "S0906",
    "text": "The notification was confirmed successfully.",
    "type": "SUCCESS"
  },
  "object": {
    "type": "message",
    "value": "1819853"
  }
} |

Push

Push is the passive retrieval of system notifications. The notifications are sent by the system to a specified address (URL) using HTTP.

Target addresses

HTTP and HTTPS addresses are possible. We recommend the use of HTTPS addresses.

Source IP of the push notification

The Source IP is 62.116.129.216.

Configure your firewall accordingly so that push notifications are not filtered out.
Hints concerning the successful delivery of a push notification

Notifications are considered as "successfully delivered" if the server sends an HTTP Status 200 response.

If a URL is defined as forwarding, it will be resolved.

If a notification cannot be sent within 24 hours, it will be delivered to the fallback email address (reply_to from the request).

⚠️ Please note that push messages cannot be delivered via email redirect.

Example PHP script for writing the push message to a textfile

This script will process the PUSH Message and write it to output.txt. Define this Script (URL+Script) in the User Profile under the API settings.

```php
<?php
// Helper methods
function getHeaderList() {
    $headerList = array();
    foreach ($_SERVER as $name => $value) {
        if (preg_match('/^HTTP_/',$name)) {
            $name = strtr(strtr($name,5,'_',' '),' ','-');
            $name = ucwords(strtolower($name));
            $value = strtr($value,' ','-');
            // add to list
            $headerList[$name] = $value;
        }
    }
    return $headerList;
}

// Variables
$file = "output.txt";

// Fetch headers
$headers = "";
foreach (getHeaderList() as $name => $value) {
    $headers .= $name . ': ' . $value . "\n";
}

// Fetch body
$body = file_get_contents('php://input');

// Fetch current date
$now = date("Y-m-d H:i:s", time());
$content = "Date: $now\nHeaders: $headers\nBody: $body\n----------------------------------\n";

// Write to file
file_put_contents($file, $content, FILE_APPEND);
?>

Email

Notifications are sent by email (default setting).