PURESEND API Docs

Date: February 10, 2020 # Home

PURESEND API Docs

Overview

- This is the Puresend Web Service API that enables clients to `send emails`, `manage lists` and `messages` via a web service interface.
- This document assumes you will be using `Java` client code to connect with the Web Service.

Security

- All of Puresend's servers use secure HTTP (HTTPS), consequently the client will need to locally install the `Puresend public SSL certificate` for the Web Server they are connecting to. See Setup SSL Certificate
- Puresend Support will need to set `PURESEND_WEB_SERVICE` Permission for an acccount before the Web Service functionality is activated.

Connecting to Web Service

- You can obtain the WSDL at:
  The endpoint is defined at the bottom of this WSDL document.

Sending Emails

- In order to send an email you will need to have a `List`, a `Message` and a `Job`. You will then create a job and start it.

Set Up SSL Certificate

Setup SSL Certificate

Before you can make calls to the Puresend Web Service you will need to import the Puresend server's SSL certificate into your JVM keystore. To do this, follow the following steps:

- Go to the Puresend Server you will be connecting to.
  - `https://psip.puresend.com/`
- Select the lock icon at the bottom of the browser.
  - A window will display the certificate information.
  - Find the Export certificate option and save the certificate locally.
  - Save as: `psip.puresend.com.crt`
- Go to your Java bin directory
- Add the certificate to Java keystore (cacerts file) using the java keytool command:
  - keytool -import -alias puresend-psip -keystore <jre path>\lib\security\cacerts-file <path to saved psip.puresend.com.crt file>
  - sudo keytool -import -alias puresend-psip -keystore <jre path>\lib\security\cacerts -file <path to saved psip.puresend.com.crt file>

Set Up Client Code

How to generate a SOAP client code?

- To generate a SOAP client from a WSDL document, please use the following open-souce tools:
- PHP User:
  1. Download wsdl2php https://github.com/rquadling/wsdl2php
  2. Extract the package
  3. Use the command tool - wsdl2php {Purecast WSDL URL}
- Java User:
  1. Download wsdl2java http://cxf.apache.org/docs/wsdl-to-java.html
  2. Extract the package
  3. Use the command tool - wsdl2java {Purecast WSDL URL}

Create a Job

Creating a job.

Create a draft job via the Web Service. You can then update this draft job with the necessary information.

Function

- `createJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobName: `String` * Job name. Must be a unique name.
- categoryId: `Integer` * Category to create job in. * `-1` for default top level category.

Return

- `int`:
  - `Job ID`: The job id of the newly created jo

Exception

- Exception

Documentation

- To create a Job via the Web Service, use this function:`createJob()`
- To update the necessary information for the Job via the Web Service, use function: `updateJob()`

Examples

```
public void createJobSample() throws Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        String jobName = "Enter Non Existing Name";
```

```
        // Use default top level category
        Integer categoryId = -1;

        Integer jobId = port.createJob(username, password, jobName, categoryId);

        if(jobId > 0) {
            System.out.println("Job has been created with id:" + jobId);
            // use updateJob() to update the Job.
        }

}
```

```
<?php
$url='https://hed-ui.puresend.com/services/Puresend?wsdl';

$params = <<<EOT
         <ns1:createJob>
             <ns1:username>XXX</ns1:username>
             <ns1:password>XXX</ns1:password>
             <ns1:jobName>Enter Non Existing Name</ns1:jobName>
             <ns1:categoryId>-1</ns1:categoryId>
         </ns1:createJob>
EOT;

$sopaVar = new SoapVar($params, XSD_ANYXML);

$client = new SoapClient($url);


$res = $client->createJob($sopaVar);


var_dump($res);


?>
```

Create Jobs by DomainThrottles

Create domain throttles Job

Create Job by domain throttles. You can then update this draft job with the necessary information.

Function

- `createDomainThrottlesJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- categoryId: `Integer` * Category to create job in. * `-1` for default top level category.
- jobName: `String` * Job name. Must be a unique name.
- Job jobInfo: `Job` * the Job object that contains the updated data

---

Return

- `List<Integer>`:
  - `Job Id array.`

Exception

- Exception

Documentation

- Create Job by domain throttles. You can then update this draft job with the necessary information.

Examples

```
public void createDomainThrottlesJobSample() {
    try {
        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
```

```
        String password = "XXX";

        Job job = new Job();

        ObjectFactory factory = new ObjectFactory();

        job.setName(factory.createJobName("TestSampleName"));
        job.getDomainThrottles().add("gmail.com,40,5,20,5,0");
        job.getDomainThrottles().add("yahoo.com,40,5,20,5,0");
        job.setRulesetId(1177);
        job.setFrom(factory.createJobFrom("aaa@aaa.com"));
        job.getListIds().add(9006);
        job.setMessageId(17253);
        job.setSubject(factory.createJobSubject("a subject"));
        job.getClusterNames().add("sampleCluster");

        port.createDomainThrottlesJob(username, password, "TestSample", -1, job);
        }
    } catch (Exception e) {
        System.out.println("An unexpected error has occurred." + e.getMessage());
    }
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();
$createDomainThrottlesJobData = new createDomainThrottlesJob();
$createDomainThrottlesJobData->username = 'XXX';
$createDomainThrottlesJobData->password = 'XXX';
$createDomainThrottlesJobData->jobName = 'TestSampleName';
$createDomainThrottlesJobData->categoryId = -1;

$jobInfoData = new Job();
$jobInfoData->name = 'TestPhp';
$jobInfoData->rulesetId = 1177;
$jobInfoData->from = 'aaa@aaa.com';
$jobInfoData->messageId = 17253;
$jobInfoData->subject = 'Test subject';
$jobInfoData->clusterNames = 'sampleCluster';
$jobInfoData->domainThrottles = "gmail.com,40,5,20,5,0";
$jobInfoData->listIds = 9006;

$createDomainThrottlesJobData->jobInfo = $jobInfoData;
```

```php
$response = $p->createDomainThrottlesJob($createDomainThrottlesJobData);
var_dump($response);

?>
```

Create Job From an existing Job

Creating a job from an existing job.

> Creating a draft job from an existing job. You can then update this draft job with the necessary information.

Function

- `createJobFromExistingJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- existingJobId: `int` * Job id of an existing job.
- newJobName: `String` * Job name for the new draft job.

Return

- `int`:
  - `Job ID`: The job id of the newly created jo

Exception

- Exception

Documentation

- Creating a draft job from an existing job. You can then update this draft job with the necessary information.
- To create a draft job from an existing job via the Web Service, use this function:`createJobFromExistingJob()`
- To update the necessary information for the Job via the Web Service, use function: `updateJob()`

Examples

```
public void createJobFromExistingJobSample() throws Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        String newJobName = "Enter Non Existing Name";

        // Use default top level category
        Integer categoryId = -1;

        Integer existingJobId = 1234;

        Integer jobId = port.createJobFromExistingJob(username, password, existingJobId, new

        if(jobId > 0) {
            System.out.println("Job has been created with id:" + jobId);
            // use updateJob() to update the Job.
        }

}
```

```
<?php

include 'puresend.php';

$p = new Puresend();

$createJobFromExistingJobData = new createJobFromExistingJob();
$createJobFromExistingJobData->username = 'XXX';
$createJobFromExistingJobData->password = 'XXX';
$createJobFromExistingJobData->newJobName = 'Enter Non Existing Name';
```

```php
$createJobFromExistingJobData->existingJobId = 1234;

$response = $p->createJobFromExistingJob($createJobFromExistingJobData);
var_dump($response);

?>
```

Abort a Job

Abort a Job.

Abort the Job by ID.

Function

- `abortJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `Integer` * The Job ID.

Return

- `int`:
  - `1`: Successfully Aborted this Job
  - `0`: Failed Aborted this Job

Exception

- Exception

Documentation

- Ability to abort the job when it is:
  - Paused,
  - NOT finished
  - the state equals Ready.

Examples

```
public void abortJobSample() throws Exception {

        String username = "XXX";
        String password = "XXX";
        int jobId = 1234;

        int result = port.abortJob(username, password, jobId);

        if (result == 1) {
            System.out.println("success");
        }

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$abortJobData = new abortJob();
$abortJobData->username = 'XXX';
$abortJobData->password = 'XXX';
$abortJobData->jobId = 1234;

$response = $p->abortJob($abortJobData);
var_dump($response);
?>
```

Delete a Job

Delete a job.

Delete the Job by ID.

Function

- `deleteJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `Integer` * The Job ID.

Return

- `int`:
  - `1`: Successfully deleted this Job
  - `0`: Failed deleted this Job

Exception

- java.lang.Exception

Documentation

- To delete a Job via the Web Service, use the function:`deleteJob()`

Examples

```
public void deleteJobSample() {
        String username = "XXX";
        String password = "XXX";
        int jobId = 1234;

        int result = port.deleteJob(username, password, jobId);

        if (result == 1) {
            System.out.println("success");
        }
}
```

```php
<?php
include 'puresend.php';

$p = new Puresend();

$deleteJobData = new deleteJob();
$deleteJobData->username = 'XXX';
$deleteJobData->password = 'XXX';
$deleteJobData->jobId = 1234;

$response = $p->deleteJob($deleteJobData);
var_dump($response);

?>
```

Get a Job Information

Get a job.

Obtain the Job information.

Function

- `getJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `Integer` * Job Id

Return

- `Job`:
  - `Job information object`
  - Returns `null` if job does not exist.

Exception

- Exception

Documentation

- To get the Job information via the Web Service, use the function:`getJob()`

Examples

```
public void getJobSample() {
        String username = "XXX";
        String password = "XXX";

        int jobId = 1234;

        Job job = port.getJob(username, password, jobId);

        if(job != null) {
            System.out.println(job.getId());
        }
}
```

```
<?php

include 'puresend.php';

$p = new Puresend();
$getJobData = new getJob();
$getJobData->username = 'XXX';
$getJobData->password = 'XXX';
$getJobData->jobId = '1234';

$jobInfo = $p->getJob($getJobData);

var_dump($jobInfo);

?>
```

Start a Job

Start a job.

Start a job by Id.

Function

- `startJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `Integer` * Job Id.

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Start a job from a draft job based on user privileges, use the function:`startJob()`

Examples

```
public void startJobSample() {
    try {
            String username = "XXX";
            String password = "XXX";

            int jobId = 1234;
```

```
            int result = port.startJob(username, password, jobId);
        }
    } catch (Exception e) {
        System.out.println("An unexpected error has occurred." + e.getMessage());
    }
}
```

```
<?php

include 'puresend.php';

$p = new Puresend();

$startJobData = new startJob();
$startJobData->username = 'XXX';
$startJobData->password = 'XXX';
$startJobData->jobId = 1234;

$response = $p->startJob($startJobData);
var_dump($response);

?>
```

Update a Job

Update a job.

Update the job information.

Function

- `updateJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobInfo: `Job` * Job object.

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Update a newly created job. use the function:`startJob()`.
- Can update the job as long as it is in the `draft status`.
- Cannot update the job once it has started.
- Job data that can be updated using the Job object is as follows, use at most:
  - list ids,
  - suppression ids,
  - cluster names,
  - rule set,
  - message id,
  - friendly from,
  - from,
  - subject,
  - scheduled date,
  - Friendly Reply-To,
  - Reply-To,
  - From format,
  - search criteria ID,
  - Skip the first,

Examples

```
public void updateJobSample() {
        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();
        ObjectFactory factory = new ObjectFactory();

        String username = "XXX";
        String password = "XXX";
```

```
        Job jobInfo = new Job();

        // Use existing Job ID
        jobInfo.setId(43196);

        jobInfo.getClusterNames().add("clusterA");
        jobInfo.setFrom(factory.createJobFrom("test@test.com"));
        jobInfo.setFriendlyFrom(factory.createJobFriendlyFrom("Enter Value For From"));

        jobInfo.getListIds().add(123);
        jobInfo.getListIds().add(234);
        jobInfo.getListIds().add(456);

        // Use existing message ID
        jobInfo.setMessageId(1234);

        // Use ruleset id
        jobInfo.setRulesetId(2345);

        // Set Job Name
        jobInfo.setName(factory.createJobName("Enter Non Existing Name"));

        // Ignore or Set to null to schedule job immediately
        Calendar cal = Calendar.getInstance();
        cal.set(2018, 7, 20);
        GregorianCalendar gcal = new GregorianCalendar();
        gcal.setTime(cal.getTime());
        jobInfo.setScheduledDate(factory.createJobScheduledDate(DatatypeFactory.newInstance(

        int result = port.updateJob(username, password, jobInfo);

        if (result == 1) {
            System.out.println("Job has been updated " + result);
        }
}
```

```
<?php

include 'puresend.php';

$p = new Puresend();

$updateJobData = new updateJob();
$updateJobData->username = 'XXX';
$updateJobData->password = 'XXX';
```

```php
$jobInfoData = new Job();
$jobInfoData->id = 43196;
$jobInfoData->from = 'newfrom@newdomain.com';

$updateJobData->jobInfo = $jobInfoData;

$response = $p->updateJob($updateJobData);
var_dump($response);

?>
```

Unpause a Job

Un-pause a Job.

Un-pause a Job by Id.

Function

- `unpauseJob()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- jobId: `Integer`
  - Job Id.

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Start a paused job based on user privileges.

Examples

```
public void unpauseJobSample() {

        String username = "XXX";
        String password = "XXX";
        int jobId = 1234;

        int result = port.unpauseJob(username, password, jobId);

        if (result == 1) {
            System.out.println("success");
        }
}
```

```
<?php

include 'puresend.php';

$p = new Puresend();

$unpauseJobData = new unpauseJob();
$unpauseJobData->username = 'XXX';
$unpauseJobData->password = 'XXX';
$unpauseJobData->jobId = 1234;

$response = $p->unpauseJob($unpauseJob);
var_dump($response);

?>
```

Pause a Job

Pause a Job.

Pause a Job by Id.

Function

- `pauseJob()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- jobId: `Integer`
  - Job Id.

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Pause a job if:
  - the job has not failed previously,
  - the job is not a draft job and has not finished.

Examples

```
public void pauseJobSample() {

        String username = "XXX";
        String password = "XXX";

        int jobId = 1234;

        int result = port.pauseJob(username, password, jobId);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$pauseJobData = new pauseJob();
$pauseJobData->username = 'XXX';
$pauseJobData->password = 'XXX';
$pauseJobData->jobId = 1234;

$response = $p->pauseJob($pauseJobData);
var_dump($response);
?>
```

Update a PureTest Job

Update PureTest Job.

Update a PureTest(A/B testing) job.

Function

- `updatePureTestJob()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- puretestJobInfo: `PuretestJob` * PuretestJob object.

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Update a newly created Pureest(A/B testing) job.
- Can update the job as long as it is in the `draft status`.
- Cannot update the job once it has started.
- Puretest Job data that can be updated using the Job object is as follows:
  - list ids,
  - suppression ids,
  - cluster names,
  - rule set,
  - message id,
  - friendly from,
  - from,
  - subject,
  - scheduled date.

Examples

```
public void updatePureTestJobSample() {

        String username = "XXX";
        String password = "XXX";

        PuretestJob puretestJobInfo = new PuretestJob();
        puretestJobInfo.setId(1234);
        puretestJobInfo.setMessageId(12345);

        int result = port.updatePuretestJob(username, password, puretestJobInfo);
}
```

```
<?php
```

```php
include 'puresend.php';

$p = new Puresend();

$updatePuretestJobData = new updatePuretestJob();
$updatePuretestJobData->username = 'XXX';
$updatePuretestJobData->password = 'XXX';

$puretestJobData = new PuretestJob();
$puretestJobData->id = 1234;
$puretestJobData->messageId = 12345;

$updatePuretestJobData->puretestJobInfo = $puretestJobData;

$response = $p->updatePuretestJob($updatePuretestJob);
var_dump($response);


?>
```

Test Filtering

Perform an email filtering test.

Perform an email filtering test.

Function

- `testFiltering()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `int` * Job ID.

Return

- `String`:
  - Filtering test result in XML format

Exception

- Exception

Documentation

- Perform an email filtering test.

Examples

```
public void testFilteringSample() {
        String username = "XXX";
        String password = "XXX";

        int jobId = 1234;

        String result = port.testFiltering(username, password, jobId);
}
```

```
<?php

include 'puresend.php';

$p = new Puresend();

$testFilteringData = new testFiltering();
$testFilteringData->username = 'XXX';
$testFilteringData->password = 'XXX';
$testFilteringData->jobId = 1234;

$response = $p->testFiltering($testFilteringData);
var_dump($response);

?>
```

Send Test Job to Proof Lists

Send Test Job to Proof Lists.

Send a test job to the specified proof list(s)

Function

- `sendTestJobToProofLists()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `int` * The test Job ID.
- proofListIds: `int[]` * id of proof list to send test job to. Ensure toEmails is set to `null`.

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Send a test job to the specified proof list(s).

Examples

```
public void sendTestJobToProofListsSample() {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
```

```
        int jobId = 1234;
        List<Integer> proofListIds = new ArrayList<>();
        proofListIds.add(123);
        proofListIds.add(234);

        int result = port.sendTestJobToProofLists(username, password, jobId, proofListIds);
}
```

```
<?php

include 'puresend.php';

$p = new Puresend();

$sendTestJobToProofListsData = new sendTestJobToProofLists();
$sendTestJobToProofListsData->username = 'XXX';
$sendTestJobToProofListsData->password = 'XXX';
$sendTestJobToProofListsData->jobId = 1234;
$sendTestJobToProofListsData->proofListIds = 123;

$response = $p->sendTestJobToProofLists($sendTestJobToProofListsData);
var_dump($response);

?>
```

Send Test Job to Emails

Send Test Job to Emails.

Send a test job to the specified emails.

Function

- `sendTestJobToEmails()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `int` * The test Job ID.
- toEmails: `String[]` * array of emails..

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Send a test job to the specified emails.

Examples

```
public void sendTestJobToEmailsSample() {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        int jobId = 1234;
        List<String> toEmails = new ArrayList<>();
        toEmails.add("email1@domain.com");
        toEmails.add("email2@domain.com");

        int result = port.sendTestJobToEmails(username, password, jobId, toEmails);
}
```

```
<?php

include 'puresend.php';

$p = new Puresend();

$sendTestJobToEmailsData = new sendTestJobToEmails();
```

```php
$sendTestJobToEmailsData->username = 'XXX';
$sendTestJobToEmailsData->password = 'XXX';
$sendTestJobToEmailsData->jobId = 1234;
$sendTestJobToEmailsData->toEmails = 'email1@domain.com';

$response = $p->sendTestJobToEmails($sendTestJobToEmailsData);
var_dump($response);

?>
```

Find Multiple Job IDs

Find JobIds.

Find a list of job IDs by name.

Function

- `findJobIds()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- nameFilter: `String` * search string
- exactMatch: `boolean` * set exact match or not.

Return

- `int[]`:
  - job IDs array

Exception

- java.lang.Exception

Documentation

- Retrieves a list of job IDs by name.
- Supports wildcard name searches.

Examples

```
public void findJobIdsSample() {
        String username = "XXX";
        String password = "XXX";
        String nameFilter = "A Name";
        boolean exactMatch = false;

        int[] jobIds = port.findJobIds(username, password, nameFilter, exactMatch);

        if (jobIds.length > 0) {
            System.out.println("success");
        }
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$findJobIdsData = new findJobIds();
$findJobIdsData->username = 'XXX';
$findJobIdsData->password = 'XXX';
$findJobIdsData->nameFilter = 'A Name';
$findJobIdsData->exactMatch = false;

$response = $p->findJobIds($findJobIdsData);
var_dump($response);

?>
```

Find Job IDs by Job Tag

Find JobIds by Tags.

Find a list of job IDs by tags.

Function

- `searchJobByTag()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- tag: `String` * search tag.
- limit: `int` * constrains the number of return job ids.
- offset: `int` * skips the offset job ids.

Return

- `int[]`:
  - job IDs array

Exception

- java.lang.Exception

Documentation

- Retrieves a list of job IDs by tag.

Examples

```
public void searchJobByTagSample() {
        String username = "XXX";
        String password = "XXX";
        String tag = "simpletag";

        int[] jobIds = port.searchJobByTag(username, password, tag, 10, 0);

        if (jobIds.length > 0) {
            System.out.println("success");
        }
}
```

```php
<?php
include 'puresend.php';

$p = new Puresend();

$searchJobByTagData = new searchJobByTag();
$searchJobByTagData->username = 'XXX';
$searchJobByTagData->password = 'XXX';
$searchJobByTagData->tag = 'simpletag';
$searchJobByTagData->limit = 10;
$searchJobByTagData->offset = 0;

$response = $p->searchJobByTag($searchJobByTagData);
var_dump($response);

?>
```

Create a List

Creating a List

Create a List via the Web Service.

Function

- `createList()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- listname: `String` * name of the list.
- brand: `String` * any human readable name used for list purpose
- isProofList: `boolean` * is this list a proof list
- fieldNames: `String[]` * custom field names for the list. Maximum 25 fields allowed.

Return

- `int`:

- `List ID`: The new List's list ID
- `0`: create Failed

Exception

- PuresendException

Documentation

- If you are using custom field names, do not specify 'Email' as this is already defined by Puresend and has its on setter function.
- Passing in 'null' for the field names will use the default Puresend list master info.

Examples

```
public void createListSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        String listName = "Enter Non Existing Name";
        String brandName = "Brand Name";
        boolean isProofList = false;
        List<String> listFieldValues = new ArrayList<>();
        listFieldValues.add("FirstName");
        listFieldValues.add("LastName");
        listFieldValues.add("Address");

        int listId = port.createList(username, password, listName, brandName, isProofList, l

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();
```

```php
$createListData = new createList();
$createListData->username = 'XXX';
$createListData->password = 'XXX';
$createListData->listname = "Enter Non Existing Name";
$createListData->brandname = "Brand Name";
$createListData->isProofList = false;
$createListData->fieldNames = array("FirstName", "LastName", "Address");

$response = $p->createList($createListData);
var_dump($response);

?>
```

Import to List via FTP

Import To List Via FTP

Imports the CSV list data with the auto import XML file to the user's FTP directory.

Function

- `importToListViaFTP()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- csvFileName: `String`
  - filename used for import. Filename used for `.xml` and `.go` file.
- csvFileData: `byte[]`
  - CSV file content data
- xmlControlFileData: `byte[]`
  - XML control data.
  - see Automated_Upload_Facility
- fieldNames: `String`
  - custom field names for the list. Maximum 25 fields allowed.

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Imports the CSV list data with the auto import XML file to the user's FTP directory.
- Ensure the account's FTP directory has been setup.

Examples

```
public void importToListViaFtpSample() throws java.lang.Exception {

        String username = "XXX";
        String password = "XXX";
        String csvFileName = "theNameOfCsv.csv";

        File csvFile = new File("csvFile.csv");
        byte[] csvFileData = Files.readAllBytes(csvFile.toPath());

        File xmlControlFile = new File("xmlControlFile.xml");
        byte[] xmlControlFileData = Files.readAllBytes(xmlControlFile.toPath());

        int result = port.importToListViaFTP(username, password, csvFileName, csvFileData, x

}
```

```
<?php


?>
```

Automated_Upload_Facility

Instructions for the Automated Import feature

Files

- You will need `three files`, all of which should be placed into your regular ftp directory:
  1. `CSV`: A standard .csv-formatted text file containing your data. The file can have any name. As usual, comma-delimit the fields, double-quote any fields containing commas or single-quotes, etc. You may include an optional header row, though any such row will be ignored for the purposes of the automated import.
  2. `XML`: A control file (again, with any filename) but with an .xml extension. This file describes the actions to be performed.
  3. `GO`: A "go" file, which can be empty, with the same name as the .xml file but with a .go extension. The import processor will not begin to process the .xml control file until this file is present. This allows you to upload your .csv and .xml in any order, and also to allow for timing delays while transmitting the .xml file. In any case, be sure to write this file last, when you are ready for your import to run. The contents of the .go file are irrelevant.

Control File Format

- The general format of the xml control file is as follows. It is not validated against an external DTD explicitly, but if the file is not well-formed or if it cannot be parsed correctly, you will receive the error message (in an email) and the file will be renamed with an "-error" extension.
- Please note that all the required text below is case sensitive. So, "Filename" must be "Filename" and not "filename", and "Delete ALL" must be "Delete ALL" and not "delete all", etc.

```
<Import>
    <Filename hasHeader="true">yourfilename.csv</Filename>
    <Notifications>susan@xyz.com,george@xyz.com</Notifications>
    <ImportType>Update</ImportType>
    <UpdateType skipUnmatched="false">Update all</UpdateType>
    <TargetList>123</TargetList>
    <CSVFields>
        <Email/>
        <First/>
        <Last/>
    </CSVFields>
    <MatchFields>
       <Email/>
    </MatchFields>
</Import>
```


- `<Filename>`: Name of a .csv file in the same directory as the .xml file. hasHeader will be "true" if the file contains a header row (which will then be skipped).
- `<Notifications>`: List of people to notify upon start and completion of the import, with commas between email addresses. Each recipient specified will receive both error and success notifications.
- `<ImportType>`: The type of import to run. Must be one of the following types:
  - *Replace* - Replace the entire target list with the contents of the csv file
  - *Append* - Add the entries in the csv file to the end of the existing list
  - *Update* - Update matching entries in the target list with the new data from the .csv file
  - *Update Broadcasts* - Update matching entries in all broadcast lists with the new data from the .csv file
  - *Delete* - Delete any entries from the target list matching any entry in the .csv file. (Special permission required.)
  - *Delete ALL* - Delete entries from all broadcast lists on the entire server matching any entry in the .csv file.(Special permission is required for this.)
- `<UpdateType>`: (Required only for an ImportType of "Update") Specifies what to do about duplicate matching entries when the update is run. The "skipUnmatched" attribute will be "true" if you want to completely skip all rows in the csv file that do not match the target list. If you set this value to "false", any non-matching rows in the .csv file will be added as new entries. The value for the UpdateType tag must be one of the following:
  - `Update one` - Update just one of the matching entries with the new data
  - `Update all` - Update all matching entries with the new data
  - `Update one and remove remaining`- Update a single entry and remove any matching duplicates.
- `<TargetList>`: The ID of the list against which to import. For an Update Broadcasts or a Delete ALL import, you can specify any valid list, otherwise use the appropriate target list ID, which is displayed on your Lists page. Updates will modify this list, appends will add to this list, deletes will delete from this list.
- `<CSVFields>`: List of all fields in your csv file, in the column order in which they appear. This might be a single field or many. They should be specified as `<FieldName/>` in the list. The field name choices (case sensitive) for imports are:

```
City
Company
```



```
Country
Email
Fax
Phone
FirstName
LastName
OptIn
Source
State
Street
Title
Zip
Custom1
Custom2
Custom3
Custom4
Custom5
```

- `<MatchFields>`: (Required only for updates and deletes) For updates and deletes, you must include one or more fields to be matched against the target list. In order for the update or delete to be performed on an existing record, the record must exactly match all of the fields that you specify here. The choices for field names are the same as for the `<CSVFields>` tag.

Example

Suppose that you wished to delete all entries from all lists matching a list of email addresses in a file. You might create a `deletethese.csv` file looking like this:

```
email1@somedomain.com
email2@somedomain.com
email3@somedomain.com
```

and so forth. Then, in the same directory, you'd put a control file, `mydelete.xml`:

```
<Import>
    <Filename hasHeader="false">deletethese.csv</Filename>
    <Notifications>robert@xyz.com</Notifications>
    <ImportType>Delete ALL</ImportType>
    <TargetList>23</TargetList>
    <CSVFields><Email/></CSVFields>
    <MatchFields><Email/></MatchFields>
</Import>
```

Once this file is written, you would write an empty file named `mydelete.go` to the same directory, and your import would be pro-

cessed (the `.xml` and `.go` files deleted after processing, or renamed to error), and the import run at the next available opportunity.

Advanced Features

Automatic CSV filename selection

- The CSV file may be automatically determined by the file name of the XML file, if desired. To do this, use this form for the Filename entry in the XML control file: `<Filename sameAsXML="true" newExtension=".csv"/>` The "`newExtension`" attribute may be omitted, in which case it will default to .csv. When this form of the `<Filename>`is used, the filename for the CSV file will be the same as the XML file, minus the `.xml` ending, and adding the ending from the "`newExtension`" in its place.
- As always, the hasHeader attribute may be included if desired: `<Filename hasHeader="true" sameAsXML="true" newExtension=".csv"/>`

CSV file renaming

- An attribute, `datestampCSV="true"`, may be added to the `<Filename>`tag to have the CSV file automatically renamed upon the processing of the XML control file. The CSV file will be renamed to have a suffix of the current date and time. For example: `someimport.csv` will become: `someimport.csv_2018-8-20_17-34-18` and that will be the file that will ultimately be imported. This attribute may be used in conjunction with automated CSV filename selection above, e.g.: `<Filename sameAsXML="true" newExtension=".csv" datestampCSV="true"/>` or with the more ordinary Filename usage: `<Filename datestampCSV="true">someimport.csv</Filename>` As always, the hasHeader attribute may be included if desired: `<Filename hasHeader="true" datestampCSV="true">someimport.csv</Filename>`
- Note that if the processing of the XML file fails for any reason, the CSV file will NOT be renamed.

Automatic target list selection

- The target list to be imported may alternatively be specified by a brand based upon the filename of the import (.csv) file.
- The first part of the filename, up to a specified delimiter, will be used as a brand.
- Then, all lists in the account will be searched for that brand (in a case-insensitive manner), and the first list found with that (case-insensitive, but otherwise exact) brand will be used as the target list of the import.
- The syntax for specifying this is: `<TargetList findBrandDelimiter="_">` where, of course, the "`_`" delimiter specified may actually be any delimiter you wish to specify (even a string of several characters).

- In this example, if the filename to be imported is: `thisbrand_tuesday_import-topexi.csv` then "thisbrand" would be searched for in the brands of the lists in the account, and the first matching list (lowest List ID) will be used as the target list for the import. If no matching list is found, then the import will fail.
- This feature may be used with the automated CSV filename selection feature described above.

Note on Timing

- By default, the automated import processor checks for these files every 15 minutes, so it might take a few minutes for your new import request to be processed.
- Also, imports generally run one at a time to avoid overloading the system, so your requested import could be queued up behind earlier imports.

(END)

Get List Id

Get List ID

Get List ID by List Name.

Function

- `getListId()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listname: `String`
  - list name to be searched.

Return

- `int`:
  - list id corresponding to the list name

Exception

- Exception

Documentation

- Retrieves the list id for the given list name.
- If multiple lists exist with the same name, the first list's id is returned.

Examples

```
public void getListIdSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        String listName = "A List Name";

        int listId = ppt.getListId(username, password, listName);

        System.out.println(listId);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getListIdData = new getListId();
$getListIdData->username = 'XXX';
$getListIdData->password = 'XXX';
$getListIdData->listname = 'A List Name';

$response = $p->getListId($getListIdData);
var_dump($response);

?>
```

Get List Info

Get List Info

Get List Info by List ID.

Function

- `getListInfo()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listId: `int`
  - list id for which data will be returned.

Return

- `ListInfo`:
  - ListInfo object containing the list information.

Exception

- Exception

Documentation

- Get List Info by List ID.

Examples

```
public void getListInfoSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();
```

```
        String username = "XXX";
        String password = "XXX";
        int listId = 123456;

        ListInfo listInfo = ppt.getListInfo(username, password, listId);

        System.out.println(listInfo.getId());
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getListInfoData = new getListInfo();
$getListInfoData->username = 'XXX';
$getListInfoData->password = 'XXX';
$getListInfoData->listId = 123456;

$response = $p->getListInfo($getListInfoData);
var_dump($response);

?>
```

Get Multiple List Info

Get List Infos

Get Multiple List information by List type.

Function

- `getListInfos()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listType: `int`

  - `[OPTIONAL]` list type.
    * `1` for regular lists
    * `2` for database suppression sists
    * `3` for file-based suppression list
    * `4` for file-based domain suppression list
    * `5` for file-based username suppression list
    * `7` for file-based MD5 suppression list
    * `8` for seed list

Return

- `ListInfo[]`:
  - ListInfo array of objects containing the list information.

Exception

- Exception

Documentation

- Retrieves all the lists including the list information for the user's account.

Examples

```
public void getListInfosSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        ListInfo[] listInfos = ppt.getListInfos(username, password, 1);

        for (ListInfo listInfo : listInfos) {
            System.out.println(listInfo.getId());
        }
}
```

```
<?php
include 'puresend.php';
```

```php
$p = new Puresend();

$getListInfosData = new getListInfos();
$getListInfosData->username = 'XXX';
$getListInfosData->password = 'XXX';
$getListInfosData->listType = 1;

$response = $p->getListInfos($getListInfosData);
var_dump($response);

?>
```

Delete a List

Delete a List

Delete List by List ID.

Function

- `deleteList()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listId: `int`
  - list id to be deleted.

Return

- `int`:
  - `1`: success.
  - `0`: failed

Exception

- Exception

Documentation

- Delete List by List ID.

Examples

```
public void deleteListSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        int listId = 123456;

        int result = ppt.deleteList(username, password, listId);

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$deleteListData = new deleteList();
$deleteListData->username = 'XXX';
$deleteListData->password = 'XXX';
$deleteListData->listId = 1234;

$response = $p->deleteList($deleteListData);
var_dump($response);

?>
```

Get All List Id

Get List IDs

Get All the List ID by user's account.

Function

- `getLists()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listType: `int`
  - `[OPTIONAL]` list type.
    - `1` for regular lists
    - `2` for database suppression sists
    - `3` for file-based suppression list
    - `4` for file-based domain suppression list
    - `5` for file-based username suppression list
    - `7` for file-based MD5 suppression list
    - `8` for seed list

Return

- `int[]`:
  - list id corresponding to the list name

Exception

- Exception

Documentation

- Get All the List ID by user's account.

Examples

```
public void getListIdSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();
```

```
        String username = "XXX";
        String password = "XXX";
        String listName = "A List Name";

        int listId = ppt.getListId(username, password, listName);

        System.out.println(listId);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getListIdData = new getListId();
$getListIdData->username = 'XXX';
$getListIdData->password = 'XXX';
$getListIdData->listname = "A List Name";

$response = $p->getListId($getListIdData);
var_dump($response);

?>
```

Add Contact To List

Add Contact To List

Add an email/contact with associated fields to the specified list.

Function

- `addContactToList()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listId: `int`

  - id of the list to which the email will be added to
- email: `String`
  - email address to be added
- fieldValues: `String[]`
  - `[OPTIONAL]` array of the field values as specified for the list. This will constitute one entry in the list.
- allowDuplcates: `boolean`
  - `[OPTIONAL]` insert an email even if it exists in the list specified.
- actionCode: `int`
  - `[OPTIONAL]` - level of how to handle duplicated email.
    - `0`: failure.
    - `1`: return existing record if email existed in the list specified.

Return

- `Contact`:
  - added Contact object.

Exception

- Exception

Documentation

- Add an email/contact with associated fields to the specified list.

Examples

```
public void addContactToListSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        String listId = 1234;
        String email = "sample@email.com";
        String[] fieldValues = new String[]{"xxx", "xxx", "xxx"};
        boolean allowDuplicates = false;
        int actionCode = 0;
```

```
        Contact contact = ppt.addContactToList(username,
                                password,
                                listName,
                                email,
                                fieldValues,
                                allowDuplicates,
                                actionCode);
}
```

```php
<?php
include 'puresend.php';

$p = new Puresend();

$addContactToListData = new addContactToList();
$addContactToListData->username = 'XXX';
$addContactToListData->password = 'XXX';
$addContactToListData->listId = 1234;
$addContactToListData->email = 'sample@email.com';
$addContactToListData->fieldValues = array("xxx", "xxx", "xxx");
$addContactToListData->allowDuplicates = false;

$response = $p->addContactToList($addContactToListData);
var_dump($response);

?>
```

Add Multiple Contacts into List

Adding Emails to a List

Add multiple email/contact with associated fields to the specified list.

Function

- `addContacts()`

Parameters

- username: `String` * Account login name.

- password: `String` * Account password.
- listId: `int` * id of the list to which the email will be added to.
- emailWithfieldValues: `String[]` * Email Address and Field values, split by `comma`. * Example: `example@domain.com,field1,field2,field3` * array of the field values as specified for the list. This will constitute one entry in the list.
- allowDuplicates: `boolean` * insert an email even if it exists in the list specified.
- actionCode: `int` * action code of how to handle duplicated email. * `0`: failure. * `1`: return existing record if email existed in the list specified.

Return

- `Contact[]`:
  - The Array of Contact object that contain new contact Id.

Exception

- Exception

Documentation

- Add multiple email/contact with associated fields to the specified list.

Examples

```
public void addContactToListSample(int listId) throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        int listId = 1234;
        List<String> emailAndFieldValues = new ArrayList<>();
        emailAndFieldValues.add("email1@domain.com,FirstName,LastName,Address");
        emailAndFieldValues.add("email2@domain.com,FirstName,LastName,Address");
```

```
        Contact[] contacts = port.addContacts(username, password, listId, emailAndFieldValue
}
```

```
<?php
$url='https://<servername>/services/Puresend?wsdl';

$params = <<<EOT
         <ns1:addContacts>
             <ns1:username>XXX</ns1:username>
             <ns1:password>XXX</ns1:password>
             <ns1:listId>1234</ns1:listId>
             <ns1:emailWithfieldValues>email1@domain.com,FirstName,LastName,Address</ns1:ema
             <ns1:emailWithfieldValues>email2@domain.com</ns1:emailWithfieldValues>
             <ns1:allowDuplicates>true</ns1:allowDuplicates>
             <ns1:actionCode>0</ns1:actionCode>
         </ns1:addContacts>
EOT;

$sopaVar = new SoapVar($params, XSD_ANYXML);

$client = new SoapClient($url);


$res = $client->addContacts($sopaVar);


var_dump($res);


?>
```

Get a Contact

Get Contact

Get email/contact details by Contact ID.

Function

- `getContact()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- contactId: `long`
  - contact id details that to be retrieved.

Return

- `Contact`:
  - Contact object containing details of the email and the field values.

Exception

- Exception

Documentation

- Retrieves email/contact details for the specified contact id.

Examples

```
public void getContactSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        long contactId = 123456789L;

        Contact contact = ppt.getContact(username,
                                password,
                                contactId);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();
```

```php
$getContactData = new getContact();
$getContactData->username = 'XXX';
$getContactData->password = 'XXX';
$getContactData->contactId = 123456789;

$response = $p->getContact($getContactData);
var_dump($response);

?>
```

Get Multiple Contact

Get Multiple Contact

Get Multiple email/contact details by Contact IDs.

Function

- `getContacts()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- contactId: `long[]`
  - Array of contact id that to be retrieved.

Return

- `Contact[]`:
  - Array of Contact object containing details of the email and the field values.

Exception

- Exception

Documentation

- Retrieves email/contact detail Array for the specified contact id Array.

Examples

```
public void getContactsSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        long[] contactIds = new long[]{1234567890L, 1234567891L};

        Contact[] contacts = ppt.getContacts(username,
                                password,
                                contactIds);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getContactsData = new getContacts();
$getContactsData->username = 'XXX';
$getContactsData->password = 'XXX';
$getContactsData->contactIds = array(1234567890, 1234567891);

$response = $p->getContacts($getContactsData);
var_dump($response);

?>
```

Get a list of Category Names

Get Category Names

Get a list category names

Function

- `getCategoryNames()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- categoryIds: `int[]`
  - Array of category ids

Return

- `String[]`:
  - Array of category names

Exception

- Exception

Documentation

- Retrieves a list category names for the specified category IDs.
- Invalid category IDs will return a category name with an invalid name descriptor.
  - `invalid category. id: xxxx`

Examples

```
public void getCategoryNameSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        List<Integer> categoryIds = new ArrayList<>();
```

```
        categoryIds.add(1234);
        categoryIds.add(2345);

        String[] names = ppt.getCategoryNames(username,
                                password,
                                categoryIds);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getCategoryNamesData = new getCategoryNames();
$getCategoryNamesData->username = 'XXX';
$getCategoryNamesData->password = 'XXX';
$getCategoryNamesData->categoryIds = array(1234, 2345);

$response = $p->getCategoryNames($getCategoryNamesData);
var_dump($response);

?>
```

Delete a Contact

Delete Contact

Delete an email/contact details in a List.

Function

- `deleteContact()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listId: `int`
  - id of the list to which the email will be removed from.
- email: `String`

- email address to be removed

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Remove an email/contact related to the specified list.

Examples

```
public void deleteContact() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        int listId = 123456;
        String email = "sample@email.com";

        Contact contact = ppt.deleteContact(username,
                                password,
                                listId,
                                email);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$deleteContactData = new deleteContact();
$deleteContactData->username = 'XXX';
```

```php
$deleteContactData->password = 'XXX';
$deleteContactData->listId = 123456;
$deleteContactData->email = "sample@email.com";

$response = $p->deleteContact($deleteContactData);
var_dump($response);

?>
```

Set Contact OptOut Status

Set opt-out Status

Set opt-out status for the specified email contact.

Function

- `setContactOptOutStatus()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- contactId: `long`
  - email/contact to set the opt-out status for
- hasOptOut: `boolean`
  - whether contact has opted out (`true` = opted out)

Return

- `int`:
  - `1`: success
  - `0`: failed

Exception

- Exception

Documentation

- Set opt-out status for the specified email contact.

Examples

```
public void setContactOptOutStatusSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        long contactId = 123456789L;

        int result = ppt.setContactOptOutStatus(username,
                                password,
                                contactId,
                                false);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$setContactOptOutStatusData = new setContactOptOutStatus();
$setContactOptOutStatusData->username = 'XXX';
$setContactOptOutStatusData->password = 'XXX';
$setContactOptOutStatusData->contactId = 123456789;
$setContactOptOutStatusData->hasOptedOut = false;

$response = $p->setContactOptOutStatus($setContactOptOutStatusData);
var_dump($response);

?>
```

Find Contact IDs

Find ContactIds

Retrieves a list of contact IDs by email address.

Function

- `findContactIds()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- email: `String`
  - email search string.
- searchSubAccounts: `boolean`
  - whether to search sub accounts

Return

- `int[]`:
  - array of contact IDs

Exception

- Exception

Documentation

- Retrieves a list of contact IDs by email address.
- Option to include searching sub-accounts.

Examples

```
public void findContactIdsSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();
```

```
        String username = "XXX";
        String password = "XXX";
        String email = "sample@email.com";

        int[] contactIds = ppt.findContactIds(username,
                                password,
                                email,
                                false);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$findContactIdsData = new findContactIds();
$findContactIdsData->username = 'XXX';
$findContactIdsData->password = 'XXX';
$findContactIdsData->email = "sample@email.com";
$findContactIdsData->searchSubAccounts = false;

$response = $p->findContactIds($findContactIdsData);
var_dump($response);

?>
```

Get Categories

Get Categories

Retrieves a list of categories by category type.

Function

- `getCategories()`

Parameters

- username: `String`
  - Account login name.
- password: `String`

  - Account password.
- categoryType: `int`
  - Category Type.
    - `3` for regular messages
    - `4` for template messages
    - `5` for confirmation messages
    - `6` for landing page messages
    - `7` for jobs
    - `8` for lists
    - `9` for seed lists

Return

- `CategoryInfo[]`:
  - array of CategoryInfo Objects

Exception

- Exception

Documentation

- Retrieves a list of categories by category type.

Examples

```
public void getCategoriesSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        int result = ppt.getCategories(username,
                                password,
                                1);
}
```

```
<?php
include 'puresend.php';
```

```php
$p = new Puresend();

$getCategoriesData = new getCategories();
$getCategoriesData->username = 'XXX';
$getCategoriesData->password = 'XXX';
$getCategoriesData->categoryType = 1;

$response = $p->getCategories($getCategoriesData);
var_dump($response);

?>
```

Get Rule Sets

Get a list of rulesets.

Retrieves a list of rulesets.

Function

- `getRuleSets()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.

Return

- `RuleSetInfo[]`:
  - array of RuleSetInfo Objects

Exception

- Exception

Documentation

- Retrieves a list of rulesets.

Examples

```
public void getCategoriesSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        List<RuleSetInfo> info = ppt.getRuleSets(username,
                                 password);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getRuleSetsData = new getRuleSets();
$getRuleSetsData->username = 'XXX';
$getRuleSetsData->password = 'XXX';

$response = $p->getRuleSets($getRuleSetsData);
var_dump($response);

?>
```

Create Suppression List

Creates a suppression list.

Creates a suppression list with name, brand and file name information

---

Function

- `createSuppList()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listName: `String`
  - name of the list.
- brandName: `String`
  - any human readable name used for list purposes
- type: `int`
  - suppression list type
- fileName: `String`
  - if this was a file-based suppression list, filename should be presented

---

Return

- `int`:
  - new list id

Exception

- Exception

Documentation

- Creates a suppression list with name, brand and file name information

Examples

```
public void createSuppListSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        String listname = "sample list name";
```

```
        String brandName = "sample brand name";
        String fileName = "fileName";

        int listId = ppt.createSuppList(username,
                                password,
                                listname,
                                brandName,
                                1,
                                fileName);
}
```

```php
<?php
include 'puresend.php';

$p = new Puresend();

$createSuppListData = new createSuppList();
$createSuppListData->username = 'XXX';
$createSuppListData->password = 'XXX';
$createSuppListData->listname = "sample list name";
$createSuppListData->brandName = "sample brand name";
$createSuppListData->fileName = "fileName";
$createSuppListData->type = 1;

$response = $p->createSuppList($createSuppListData);
var_dump($response);

?>
```

Edit Suppression List

Edit a suppression list.

Edit a suppression list with name, brand and file name information

---

Function

- `editSuppList()`

Parameters

- username: `String`

  - Account login name.
- password: `String`
  - Account password.
- listId: `int`
  - ID of the suppression list
- listName: `String`
  - name of the list.
- brandName: `String`
  - any human readable name used for list purposes
- type: `int`
  - suppression list type
- fileName: `String`
  - if this was a file-based suppression list, filename should be presented

---

Return

- `int`:
  - existing list id

Exception

- Exception

Documentation

- Edit a suppression list with name, brand and filename information

Examples

```
public void editSuppListSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        int listId = 123456;
        String listname = "sample list name";
        String brandName = "sample brand name";
        String fileName = "fileName";
```

```
        int listId = ppt.editSuppList(username,
                                      password,
                                      listId,
                                      listname,
                                      brandName,
                                      1,
                                      fileName);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$editSuppListData = new editSuppList();
$editSuppListData->username = 'XXX';
$editSuppListData->password = 'XXX';
$editSuppListData->listname = "sample list name";
$editSuppListData->brandName = "sample brand name";
$editSuppListData->fileName = "fileName";
$editSuppListData->listId = 123456;

$response = $p->editSuppList($editSuppListData);
var_dump($response);

?>
```

Get List Entries

Get List Entries.

Retrieves email details for the specified List

Function

- `getListEntries()`

Parameters

- username: `String`
  - Account login name.

- password: `String`
  - Account password.
- listId: `int`
  - ID of the list
- offset: `int`
  - parameter can be passed to offset in the results.
- limit: `int`
  - parameters can be passed to limit the number of returned contacts

---

Return

- `String`:
  - String object containing one list recipient per line. Format of each line:
    - (Entry Id, List Id, Email, Import Date, Last Mail Date, Bounced, optout, Temp bonces, Email Format, Custom Fields size, Custom Fields(if any))

Exception

- Exception

Documentation

- Edit a suppression list with name, brand and filename information

Examples

```
public void getListEntriesSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        int listId = 123456;

        String result = ppt.getListEntries(username,
                                password,
                                listId,
```

```
                                    1,
                                    20);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getListEntriesData = new getListEntries();
$getListEntriesData->username = 'XXX';
$getListEntriesData->password = 'XXX';
$getListEntriesData->listId = 123456;
$getListEntriesData->offset = 1;
$getListEntriesData->limit = 20;


$response = $p->getListEntries($getListEntriesData);
var_dump($response);

?>
```

Update a Contact

Update a Contact

Update a Contact

Function

- `updateContact()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- contactId: `long`
  - email/contact to set the opt-out status for
- optOut: `boolean`
  - set opted out or not.

- emailBounced: `boolean`
  - set email Bounced or not.
- clearField: `boolean`
  - set clear field flag,
    - `if (clearField is true && newValue[i] is Empty)` => `clear` this Field's value,
    - `if (clearField is false)` => set those newValues which are not empty.
- newFieldValues: `String[]`
  - the new value array to set.

Return

- `int`:
  - `1`: success

Exception

- Exception

Documentation

- Update a Contact by ID.

Examples

```
public void updateContactSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        long contactId = 123456789L;
        String[] newFieldValues = new String[]{"field1", "field2"};

        int result = ppt.updateContact(username,
                                password,
                                contactId,
                                false,
```

```
                                false,
                                true,
                                newFieldValues);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$updateContactData = new updateContact();
$updateContactData->username = 'XXX';
$updateContactData->password = 'XXX';
$updateContactData->contactId = 123456789;
$updateContactData->optOut = false;
$updateContactData->emailBounced = false;
$updateContactData->clearField = true;
$updateContactData->newFieldValues = array('field1', 'field2');

$response = $p->updateContact($updateContactData);
var_dump($response);

?>
```

Update a Contact By Email

Update Contacts By Email

Update Contacts By Email

Function

- `updateContactByEmail()`

Parameters

- username: `String`
  - Account login name.
- password: `String`
  - Account password.
- listId: `int`
  - set -1 to update across All lists.

- email: `String`
  - email to set the opt-out status for.
- optOut: `boolean`
  - set opted out or not.
- emailBounced: `boolean`
  - set email Bounced or not.

Return

- `int`:
  - `1`: success

Exception

- Exception

Documentation

- Update Contact by Email.

Examples

```
public void updateContactByEmailSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType ppt = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        String email = "example@domain.com";
        int listId = -1;

        int result = ppt.updateContactByEmail(username,
                                password,
                                listId,
                                email,
                                false,
                                false);
}
```

```php
<?php
include 'puresend.php';

$p = new Puresend();

$updateContactByEmailData = new updateContactByEmail();
$updateContactByEmailData->username = 'XXX';
$updateContactByEmailData->password = 'XXX';
$updateContactByEmailData->listId = -1;
$updateContactByEmailData->email = "example@domain.com";
$updateContactByEmailData->optOut = false;
$updateContactByEmailData->emailBounced = false;

$response = $p->updateContactByEmail($updateContactByEmailData);
var_dump($response);

?>
```

Create a Message

Creating a Message

Creates a Content Message for use in a job

Function

- `createMessage()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- name: `String` * name of content message.
- categoryId: `int`
  - category to create Message in.
  - `-1` for default top level category.
- htmlContent: `String`
  - message content for html based viewing
- textContent: `String`
  - message content for text viewing allowed.
- mobileContent: `String`
  - message content for mobile viewing allowed.

- headerId: `int`
  - content ID of header
- footerId: `int`
  - content ID of footer

Return

- `int`:
  - `message ID`: the newly created message id

Exception

- Exception

Documentation

- Creates a Content Message for use in a job.

Examples

```
public void createMessageSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        String contentName = "Content Name";
        int categoryId = -1;
        String htmlContent = "htmlContent sample";
        String textContent = "textContent sample";
        String mobileContent = "mobileContent sample";
        int headerId = 123;
        int footerId = 234;

        port.createMessage(username, password, contentName, categoryId, htmlContent, textCon

}
```

```php
<?php
include 'puresend.php';

$p = new Puresend();

$createMessageData = new createMessage();
$createMessageData->username = 'XXX';
$createMessageData->password = 'XXX';
$createMessageData->categoryId = -1;
$createMessageData->htmlContent = "htmlContent sample";
$createMessageData->textContent = "textContent sample";
$createMessageData->mobileContent = "mobileContent sample";
$createMessageData->headerId = 123;
$createMessageData->footerId = 234;

$response = $p->createMessage($createMessageData);
var_dump($response);

?>
```

Create Message from URL

Create Message From URL

Create Message From URL

Function

- `createMessageFromURL()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- name: `String` * name of content message.
- categoryId: `int`
  - category to create Message in.
  - `-1` for default top level category.
- htmlContentURL: `String`
  - message content URL for html based viewing
- textContentURL: `String`
  - message content URL for text viewing allowed.

- mobileContentURL: `String`
  - message content URL for mobile viewing allowed.
- headerId: `int`
  - content ID of header
- footerId: `int`
  - content ID of footer

Return

- `int`:
  - `message ID`: the newly created message id

Exception

- PuresendException

Documentation

- Sets and post-processes the specified content type to the contents of the specified URL and creates a Message.
- Fails if the MIME type returned by the URL doesn't match the requested MIME type.

Examples

```
public void createMessageFromURLSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();
        String username = "XXX";
        String password = "XXX";
        String messageName = "SampleName";
        String htmlContentURL = "https://sample/html/content";
        String textContentURL = "https://sample/text/content";
        String mobileContentURL = "https://sample/mobile/content";
        Integer headerId = 1234;
        Integer footerId = 1235;

        int messageId = port.createMessageFromURL(username, password, messageName, -1, html(
```

```
        System.out.println(messageId);
}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$createMessageFromURLData = new createMessageFromURL();
$createMessageFromURLData->username = 'XXX';
$createMessageFromURLData->password = 'XXX';
$createMessageFromURLData->name = "SampleName";
$createMessageFromURLData->categoryId= -1;
$createMessageFromURLData->htmlContentURL = "https://sample/html/content";
$createMessageFromURLData->textContentURL = "https://sample/text/content";
$createMessageFromURLData->mobileContentURL = "https://sample/mobile/content";
$createMessageFromURLData->headerId = 1234;
$createMessageFromURLData->footerId = 1235;

$response = $p->createMessageFromURL($createMessageFromURLData);
var_dump($response);

?>
```

Get Message Information

Get Message

Get Message Information by Message ID.

Function

- `getMessage()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- messageId: `int` * the message id to be retrieved.

Return

- `Message`:
  - Message detailed information object.

Exception

- PuresendException

Documentation

- Retrieves Message details for specified message id and includes rendered data.

Examples

```java
public void createMessageFromURLSample() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";
        int messageId = 1;

        Message message = port.getMessage(username, password, messageId);

}
```

```php
<?php
include 'puresend.php';

$p = new Puresend();

$getMessageData = new getMessage();
$getMessageData->username = 'XXX';
$getMessageData->password = 'XXX';
$getMessageData->messageId = 1;

$response = $p->getMessage($getMessageData);
var_dump($response);
```

```
?>
```

Update Message based on the Mime Type.

Update Message

Updates an existing Message based on the Mime Type.

Function

- `updateMessage()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- message: `Message` * new message to replace existing message.
- mimeType: `String` * for the message version to be updated. Valid types are: * `text/html`, * `text/plain`, * `text/x-html-mobile`

Return

- `int`:
  - `1`: success.
  - `-1`: failed.

Exception

- Exception

Documentation

- Updates an existing Message based on the Mime Type.

Update Message based on the Mime Type.

Examples

```
public void updateMessage() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        // a existing Message.
        Message message = new Message();
        message.setId(1234);
        // set the message to new Category
        message.setCategoryId(12);

        String mimeType = "text/html";
        Integer headerId = 1234;
        Integer footerId = 1235;


        int result = port.updateMessage(username, password, message, mimeType, headerId, foo

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$updateMessageData = new updateMessage();
$updateMessageData->username = 'XXX';
$updateMessageData->password = 'XXX';

$messageInfoData = new Message();
$messageInfoData->id=1234;
$messageInfoData->categoryId = 12;

$updateMessageData->message = messageInfoData;

$updateMessageData->mimeType = "text/html";
$updateMessageData->headerId = 1234;
$updateMessageData->footerId = 1235;

$response = $p->updateMessage($updateMessageData);
var_dump($response);
```

```
?>
```

Update Message based from URL

Update Message From URL

Updates an existing Message with the specified URL.

Function

- `updateMessageFromURL()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- message: `Message` * new message to replace existing message.
- htmlContentURL: `String`
  - message content URL for html based viewing
- textContentURL: `String`
  - message content URL for text viewing allowed.
- mobileContentURL: `String`
  - message content URL for mobile viewing allowed.
- headerId: `int`
  - content ID of header
- footerId: `int`
  - content ID of footer

Return

- `int`:
  - `1`: success.
  - `-1`: failed.

Exception

- Exception

Documentation

- Sets and post-processes the specified content type to the contents of the specified URL and update an existing Message Object.
- Fails if the MIME type returned by the URL doesn't match the requested MIME type.

Examples

```
public void updateMessageFromURL() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        String htmlContentURL = "https://sample/html/content";
        String textContentURL = "https://sample/text/content";
        String mobileContentURL = "https://sample/mobile/content";

        int messageId = 123;

        Integer headerId = 1234;
        Integer footerId = 1235;


        int result = port.updateMessage(username, password, messageId, htmlContentURL, text(

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$updateMessageData = new updateMessage();
$updateMessageData->username = 'XXX';
$updateMessageData->password = 'XXX';
$updateMessageData->htmlContentURL = "https://sample/html/content";
$updateMessageData->textContentURL = "https://sample/text/content";
$updateMessageData->mobileContentURL = "https://sample/mobile/content";
$updateMessageData->messageId = 123;
$updateMessageData->headerId = 1234;
$updateMessageData->footerId = 1235;
```

```php
$response = $p->updateMessage($updateMessageData);
var_dump($response);

?>
```

Delete a Message

Delete Message

Deletes a Content Message.

Function

- `deleteMessage()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- message: `Message` * new message to replace existing message.
- contentId: `int` * id of the message to delete.

Return

- `int`:
  - id of the message to delete.

Exception

- Exception

Documentation

- Deletes a Content Message.
- You cannot delete a message that has been used in a job.

Examples

```
public void deleteMessage() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        int contentId = 1234;

        port.deleteMessage(username, password, contentId);

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$deleteMessageData = new deleteMessage();
$deleteMessageData->username = 'XXX';
$deleteMessageData->password = 'XXX';
$deleteMessageData->contentId = 1234;

$response = $p->deleteMessage($deleteMessageData);
var_dump($response);

?>
```

Get Job Report

Get Job Report

Get Job Report in XML formatted string.

Function

- `getJobReport()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `int`
  - Job ID

Return

- `String`:
  - Job report in XML format

Exception

- Exception

Documentation

- Returns standard job report represented as a XML formatted string.

Examples

```
public void getJobReport() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        int jobId = 123;

        String result = port.getJobReport(username, password, jobId);

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();
```

```php
$getJobReportData = new getJobReport();
$getJobReportData->username = 'XXX';
$getJobReportData->password = 'XXX';
$getJobReportData->jobId = 123;

$response = $p->getJobReport($getJobReportData);
var_dump($response);

?>
```

Get Job Report Detail

Get Job Report Detail

Get Job Report in XML formatted string.

Function

- `getJobReportDetail()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- jobId: `int`
  - Job ID
- type: `int`
  - Report type. Value:
    - `2`: Clickthru
    - `3`: Total opens
    - `4`: Opt out
    - `8`: Permanent bounce
    - `9`: Temporary bounce
    - `10`: Blocked bounce
    - `36`: Feedback loop complaints
    - `38`: Queued
    - `40`: Delivered

Return

- `String`:
  - Returns standard job report detail represented as a XML formatted string.

Exception

- Exception

Documentation

- Returns standard job report represented as a XML formatted string.

Examples

```
public void getJobReportDetail() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        int jobId = 123;

        String result = port.getJobReportDetail(username, password, jobId, 2);

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getJobReportDetailData = new getJobReportDetail();
$getJobReportDetailData->username = 'XXX';
$getJobReportDetailData->password = 'XXX';
$getJobReportDetailData->jobId = 123;
$getJobReportDetailData->type = 2;

$response = $p->getJobReportDetail($getJobReportDetailData);
```

```php
var_dump($response);

?>
```

Get Job Ids

Get Job IDs

Get multiple Job ID.

Function

- `getJobIds()`

Parameters

- username: `String` * Account login name.
- password: `String` * Account password.
- start date: `String`
  - The start date of the date range,
  - In `yyyy-MM-dd` format
- end date: `String`
  - The end date of the date range
  - In `yyyy-MM-dd` format

Return

- `int[]`:
  - Job ID array.
  - `null` if no job exist in the Date Range.

Exception

- Exception

Documentation

- Get Job ID array in the Date Range.

Examples

```
public void getJobIds() throws java.lang.Exception {

        Puresend p = new Puresend();
        PuresendPortType port = p.getPuresendHttpsSoap12Endpoint();

        String username = "XXX";
        String password = "XXX";

        intp[] jobIds = port.getJobIds(username, password, "2018-01-01", "2018-08-27");

}
```

```
<?php
include 'puresend.php';

$p = new Puresend();

$getJobIdsData = new getJobIds();
$getJobIdsData->username = 'XXX';
$getJobIdsData->password = 'XXX';
$getJobIdsData->startDate = "2018-01-01";
$getJobIdsData->endDate = "2018-08-27";

$response = $p->getJobIds($getJobIdsData);
var_dump($response);

?>
```

About

About

Puresend Web Service API Documentation

- This version 9.10 of the Purecast API Documentation was last updated on: February 10, 2020

For Additional Information

- If upon your review you have any questions or concerns, please contact us:
  - Email our support staff at `techsupport@puresend.com`
  - Call our support staff at `212-381-7374`
- If you would like to export our entire API Documentation, Click Here

Export API as a PDF

Download Puresend API PDF

Puresend Web Service API Documentation

- Download PDF