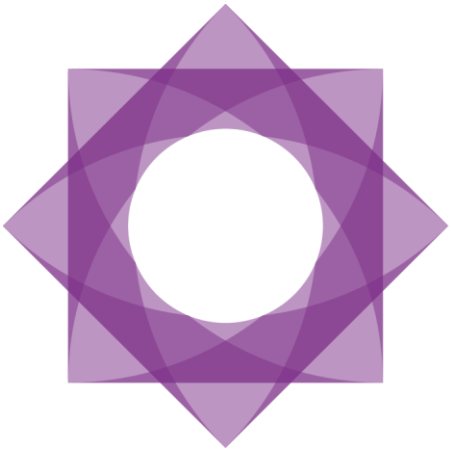


Formpipe.
Lasernet

Lasernet 9
Web Service API

Lasernet 9 – Web Service API [Revision 2 – May 2019]

Company website www.formpipe.com

Product website www.lasernetbyformpipe.com

1 INTRODUCTION

1.1 Who Should Use This Guide

This guide is written for external developers who want to gain access to queues and jobs via the Lasernet Web Service API. It is intended primarily as guide to the available API functions with options, examples and outcomes for each.

Throughout the document the reader is addressed as you, where “you” means any of the above mentioned people.

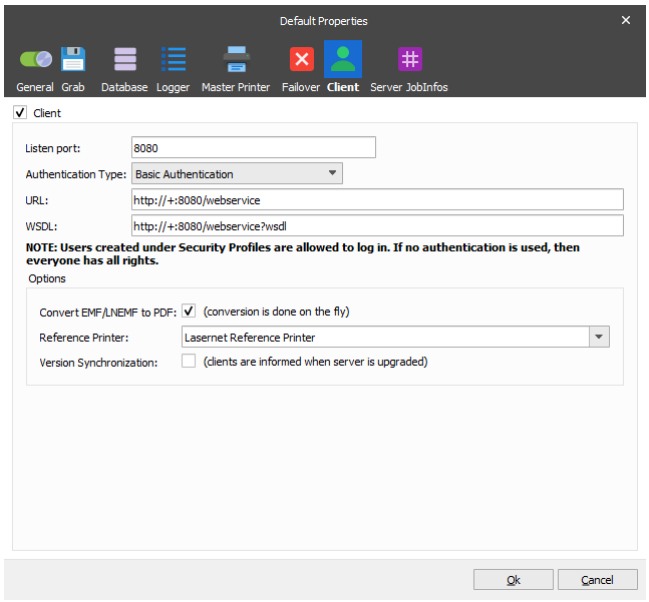
2 TERMS OF USE

3 METHODS AND PARAMETERS

3.1 How

The Web Service API is used by the Lasernet Client for access to queues and jobs in Lasernet. We recommend that you install the Lasernet Client and read the accompanying manual, to get an overview of all of the features.

The API will be accessible once the Client has been enabled in the settings of the Lasernet server.



The Web Server port supports different authentication types including Basic, SSL and a combination of Basic and SSL.

When the service is started, the following line can be seen in the log:

URL added: http://+:8080/webservice/

This means that the API can be contacted on the Lasernet server port 8080.

More info about client settings is available in the manual “Lasernet 9 – Developer”.

3.1.1 Web Services Description Language (WSDL)

A WSDL file describes the methods and their parameters which the API exposes.

If the software you wish to integrate can make web references, access to WSDL is accessible via the following URL;

http://<server>:8080/webservice?wsdl

3.2 Terms

A short introduction to the terms used in Lasernet.

3.2.1 Module

A module is an input port, an engine or an output port.

3.2.2 Queue

A Queue is an instance of a Module. There can be many Queues of the same type of Module.

Not all Queues in Lasernet are accessible through the API. Accessible Queues are those which are Paused, contains Scheduled, Failed or Completed Jobs.

3.2.3 Job

Jobs are passed between Queues. Every time a Job is passed it is cloned and given a new ID. A Job represents data at a certain time.

3.2.4 JobInfo

Every Job has metadata. Each element of metadata is called a JobInfo within Lasernet. A Job can have many JobInfos, include arrays with the index starting at 0. Some JobInfos are system specific and as such are read-only.

3.3 Methods

The API exposes a range of methods. These will be described here in detail.

3.3.1 VersionGet

Returns information about currently running version. For future use.

Returns:

VersionInfo:

Lasernet: The server version of Lasernet.

API: Version of the Lasernet API.

3.3.2 ModuleList

This method returns the names and icons of all modules in Lasernet. Each module has a unique ModuleID. These IDs are used by other methods to identify specific Modules in their results.

Parameters: None

Result: A list of ModuleInfo's.

ModuleInfo:

Name: The name of the module.

ModuleID: A GUID.

Icons: 16, 24 and 32 pixel versions of the module icons used in Lasernet.

3.3.3 JobQueuesList

This method returns a list of all queues in Lasernet of a given type. Queues are split by form types.

Parameters:

Type: Any, Input, Pause, ScheduledJobs, FailedJobs, CompletedJobs.

Result: A list of JobQueues.

JobQueue:

Name: The name of the queue.

FormType: If JobInfo 'FormType' is set on Jobs they will be grouped by this. This means the same queue can exist multiple times with different form types. The default is blank.

Count: Number of Jobs in the queue.

ModuleID: Unique ID of the queue. Please refer to ModuleList method.

Type: The 'Type' of queue: Input, Pause, ScheduledJobs, FailedJobs, CompletedJobs.

Kind: Specifies whether the queue is an Input, Engine or Output.

Rights:

Add: If it is possible to add Jobs to the queue, value will be 'True'. Please refer to JobAdd method.

3.3.4 JobQueueList

This method returns a list of Jobs in a specific queue.

Parameters:

QueueName: The name of the queue returned in JobQueuesList method.

QueueType: The type of the queue returned in JobQueuesList method.

ModuleID: The ID of the module returned in JobQueuesList method.

FormType: The form type returned in JobQueuesList method.

Keyword: Can be used to filter results. See Keyword section.

Statuses: A list of statuses to return. Default blank.

PageSize: Number of pages to divide the queue into, for example 500 or 1000 Job's.

Page: Which page to return Jobs from. See Paging section.

Result:

TotalJobs: Total number of Jobs in the Queue, regardless of PageSize. See Paging section.

List of Jobs.

Job:

JobID: Unique Lasernet ID for the Job.

Status: Status of the Job: Running, Scheduled, Delivering, Done, Cancelled, Failed, Deleted, Paused. See Status section.

Date: Date and Time of when Job was last changed.

QueueName: Name of the Queue the Job is currently in.

Schedule: The date and time for a Job if it has been scheduled for later delivery.

Action: Tells Lasernet what to do with Jobs which timed out while it was offline. Either release job at once (AtOnce) or reschedule (Reschedule) for later release.

JobData: Content of the Job itself. Only filled when the OnlyDatabase parameter is set to 'False'.

JobInfos: A list of JobInfos for the Job.

Rights: Not currently used.

3.3.5 JobQueueReleaseAsIs

Releases a whole queue. Can be filtered by a keyword (optional).

Parameters:

QueueName: The name of the queue returned in JobQueuesList method.

QueueType: The type of the queue returned in JobQueuesList method.

ModuleID: The ID of the module returned in JobQueuesList method.

FormType: The form type returned in JobQueuesList method.

Keyword: Can be used to filter results. See Keyword section.

Result:

None. Refresh the queue to see release results.

3.3.6 JobQueueReSchedule

Re-schedules all Jobs in a queue. Can be filtered by a keyword (optional).

Parameters:

QueueName: The name of the queue returned in JobQueuesList method.

QueueType: The type of the queue returned in JobQueuesList method.

ModuleID: The ID of the module returned in JobQueuesList method.

FormType: The form type returned in JobQueuesList method.

Keyword: Can be used to filter results. See Keyword section.

Schedule: When to release Job from queue.

Action: AtOnce or Reschedule. Describes what should happen to Job if Lasernet is offline at the time the scheduled release should occur. Either release the Job at once or calculate a new schedule in the future.

Result:

None. Refresh the queue to see the rescheduling results.

3.3.7 JobQueueRemove

Removes all Jobs in a queue. Can be filtered by a keyword (optional).

Parameters:

QueueName: The name of the queue returned in JobQueuesList method.

QueueType: The type of the queue returned in JobQueuesList method.

ModuleID: The ID of the module returned in JobQueuesList method.

FormType: The form type returned in JobQueuesList method.

Keyword: Can be used to filter results. See Keyword section..

Result:

None. Refresh the queue to see Job deletion result.

3.3.8 JobFind

Returns Jobs matching a specified keyword.

Parameters:

Keyword: String to search for across saved JobInfos in database.

Statuses: A list of statuses to return. Default blank.

PageSize+Page: See paging section.

Result:

Same as JobQueueList method.

3.3.9 JobAdd

This method makes it possible to add Jobs to Input and Paused queues.

Parameters:

QueueName: Which queue to add the Job in.

FileName: Name of the file in JobData including extension.

JobData: Base64 encoded Job content.

PausedUntil: Optional schedule. Will be set to AtOnce action (see JobQueueReSchedule**).**

JobInfos: List of JobInfos to add to Job.

JobInfo: Key value pairs with index starting at 0.

Result:

Newly assigned JobID. Blank if job was not created.

3.3.10 JobGet

Returns Job and JobInfos.

Parameters:

JobID: Internal JobID of the Job.

OnlyDatabase: Only return JobInfos from Database rather than from Job *and* database.

Result:

Status: Status of the Job. See Status section.

Date: Date and time of the Job.

QueueName: Which Queue the Job is in.

Schedule: If job is scheduled the date and time will be shown.

Action: Current reschedules action of the Job.

JobData: Base64 encoded content of Job

JobInfos: List of JobInfos for the Job. Key value pairs with Index starting at 0.

3.3.11 JobReschedule

This method makes it possible to reschedule a Job.

Parameters:

JobID: Unique ID of the Job.

Schedule: Date and time of the new schedule.

Action: Tells Lasernet what to do with Jobs which timed out while Lasernet was offline. Either release job at once (AtOnce) or reschedule (Reschedule) for later release.

Result:

Boolean.

3.3.12 JobDestinationsList

This method returns a list of destinations for a paused or scheduled Job.

Parameters:

JobID: Unique ID of the Job.

Result:

A list of JobDestinations.

JobDestination:

Name: Name of the destination module.

Alternative: ‘False’ if active destination. ‘True’ if destination is an optional destination.

ModuleID: ID of the destination module if it is found in configuration. See ModuleList method.

3.3.13 JobsReleaseAsIs

Releases a list of Jobs to their default destinations. Jobs can be Paused, Failed or Scheduled.

Parameters:

JobIDs: An array of JobIDs.

Result:

None. Refresh queues to see result.

3.3.14 JobRelease

Releases a paused, scheduled or failed Job. Paused Jobs will be passed to destinations specified in the Destinations parameter. Scheduled Jobs will be rescheduled to be released at once. Failed Jobs will be retried.

Parameters:

JobID: Unique ID of the Job.

Destinations: A list of destinations to pass Job to. Only used for Paused Jobs.

Result:

None.

3.3.15 JobRemove

Removes a Job from the Queue. Any schedule will be removed. If part of a combining Queue, it will be removed from this and will not be a part of the combined Job. Status of Job itself will be set as Cancelled.

Parameters:

JobID: Unique ID of the Job.

Result:

True if successful.

3.3.16 JobQueueClear

Calls JobRemove for all Jobs in a specified Queue.

Parameters:

QueueName: Name of the Queue to cancel all Jobs in.

Result:

None.

3.3.17 JobList

Returns JobInfos of a specified Job.

Parameters:

JobID: Unique ID of the Job.

OnlyDatabase: Only return JobInfos saved to database. If ‘False’, all JobInfos from Job will be returned (slow).

Result:

A list of JobInfos. Key value pair with index starting at 0.

Rights:

Edit: Set to ‘False’ if JobInfo is read-only.

3.3.18 JobAddJobInfo

Add a JobInfo to a Job both on disk and in database.

Parameters:

JobID: Unique ID of the Job to add JobInfo to.

Key: Key of the JobInfo.

Value: Value of the JobInfo

Replace: If JobInfo exists the value will be appended so JobInfo will turn into an array. If Replace is set 'True', then existing JobInfo will be overridden and an array will not be created.

Result:

Returns assigned index in array.

3.3.19 JobGetJobInfo

Returns the value for a given JobInfo.

Parameters:

JobID: Unique ID of the Job.

Key: Key of the JobInfo.

Index: Index in an array. If JobInfo is not an array, use 0.

Result:

Value of JobInfo as a string.

3.3.20 JobEditJobInfo

Edit an existing JobInfo in a Job.

Parameter:

JobID: Unique ID of the Job to edit a JobInfo for.

Key: Key of the JobInfo.

Value: Value of the JobInfo.

Index: Index of the JobInfo. Use 0 if not an array.

Result:

Returns ‘True’ if successful.

3.3.21 JobRemoveJobInfo

Removes a specific JobInfo from a Job.

Parameter:

JobID: Unique ID of the Job to remove the JobInfo from.

Key: Key of the JobInfo.

Index: Index of the JobInfo.

Result:

Returns ‘True’ if successful.

3.3.22 JobChangeJobInfoValidation

Change validation flag on JobInfo.

Parameter:

JobID: Unique ID of the Job to edit the JobInfo for.

Key: Key of the JobInfo.

Index: Index of the JobInfo. Use 0 if not an array.

Validation: None, Required, Successful, Failure.

Result:

Returns ‘True’ if successful.

3.3.23 OverlayList

Returns a list of existing overlay files in configuration.

Parameter: None

Result:

A list of filenames.

3.3.24 OverlayGet

Returns a specific overlay file either as EMF or PDF.

Parameter:

Overlay: Filename of overlay. See OverlayList.

Format: Overlay, PDF, Thumbnail.

Result:

Returns file.

3.3.25 JobPreviewGet

Returns preview data if available in Job.

Parameter:

JobID: Unique ID of the Job to get data for.

JobDataIfNoPreview: If set, then will return JobData if no PreviewJobData is found.

Result:

JobInfos: Contains just the extension JobInfo so caller knows how to display JobData.

JobData: Base64 encoded string containing file.

3.3.26 RequestComputerList

Returns list of servers in configuration.

Parameter: None

Result:

List of servers.

Computer:

Name: Instance name

Server: Server name

Port: Port open for monitor

3.3.27 RequestPrinterProfileList

Returns list of printer profiles belonging to a specific server.

Parameter:

Server: Server name. See RequestComputerList.

Port: Server port. See RequestComputerList.

Result:

List of printers.

Printer:

Name: Name of printer

Profiles:

Name

PaperSize

Width

Length

FormName

Orientation

Scale

Copies

Papersource

PrintQuality

ColorMode

DuplexMode

3.3.28 RequestModuleList

Returns list of modules belonging to aa specific server.

Parameter:

Server: Server name. See RequestComputerList.

Port: Server port. See RequestComputerList.

Result:

List of modules.

Module:

Name

ModuleID

Type

Description

Details

Inactive

3.3.29 RequestFormList

Returns a list of forms belonging to a specific module.

Parameter:

Module: Name of module in config.

Result:

List of forms.

Form:

Name

Description

Inactive

3.3.30 PhraseList

Returns a list of existing phrase objects in configuration.

Parameter: None

Result:

A list of phrase names.

3.3.31 PhraseGet

Returns a specific phrase object either as SNX, DOCX or PDF.

Parameter:

Phrase: Object name of phrase. See PhraseList.

Format: SNX, DOCX, PDF.

Result:

Returns phrase.

3.4 Keywords

The JobQueueList and JobFind methods both have filtering by keyword functionality. Filtering by keyword is done across all stored JobInfos in the database. It is not possible to filter on all JobInfos in Job – only those stored in the database by JobInfo profile.

3.5 Paging

The API uses Web Services and XML to deliver data when it is requested. To provide the most efficient use of resources, paging is used to handle the data where multiple clients are continually polling the server.

The JobQueueList and JobFind methods support returning Jobs in groups. Jobs are divided into groups of specified PageSize. The Page parameter specifies which page to return Jobs from. The first time the API is called, the specified page size should be 1. This method uses 'TotalJobs', which, when divided by PageSize and rounded up, gives the total number of pages in the Queue.

If too high a page is specified, the last page is returned.

The last page always contains PageSize number of Jobs if Queue has equal or more than PageSize Jobs, e.g.

501 Jobs in Queue. PageSize set as 500. Page 2 will not contain 1 job but rather the last 500 Jobs.

3.5.1 Method for fetching pages in Lasernet Client

The Lasernet Client fetches data in pages from the Lasernet server. When using the embedded Lasernet database (the default method for the Job Engine), only one request to the Web Service API can be made at a time. Refreshing datasets with 1000s of rows is not desirable as this will make other Clients hang and the server will stop processing jobs. Therefore paging is implemented to keep the performance of the server at a manageable level without reducing functionality.

Paging works only on date. It is not affected by how the view is sorted in the Lasernet Client. This means the first page will always contain the oldest jobs and the last page the newest jobs.

3.6 Status

A Job can only have one active status at any given time, though it may change many times as it is processed.

Running: Currently being processed. Job should be left alone.

Scheduled: Scheduled for later release. Job is essentially paused until schedule occurs.

Delivering: Currently being processed. Job should be left alone.

Done: Job has completed. Lasernet has finished processing Job. Job should be left alone.

Cancelled: Job has been cancelled. Job should be left alone.

Failed: Job has failed but problem might be fixable via JobInfo and Job can be retried.

Deleted: Job has been deleted. Job should be left alone.

Paused: Job is in a paused state requiring manual release.

Only Scheduled, Done, Failed and Paused Jobs should be considering when working with the API.