

SERFF State API Version 4.0
Web Services
Developer's Guide

© 2007 National Association of Insurance Commissioners (NAIC)

Table of Contents

INTRODUCTION	1
OVERVIEW	2
USAGE LIMITATIONS	3
SECURITY AND AUTHENTICATION	4
SHARED OBJECT DETAIL	5
EXCEPTIONS.....	6
ClientException.....	6
ServerException	6
CallRateExceededException	6
WEB SERVICES	7
Filing Services.....	7
QueryFiling.....	7
Correspondence Services.....	8
submitObjectionLetter	8
finalizeObjectionLetter	9
submitNoteToFiler.....	10
finalizeNoteToFiler.....	11
submitDisposition	12
finalizeDisposition	13
Update Filing Services	14
setReviewerName	14
setStateTrackingNumber.....	15
setStateStatus	16
Aggregate PDF Services	16
RequestPdf	16
FinalizePdf	17
Handling Attachments Services.....	18
beginUpload.....	18
writeBlock.....	19

endUpload.....	19
Download Attachment Handling Services.....	20
The Download Attachment Handling web services greatly facilitate downloads with large file attachments.....	20
beginDownload.....	20
readBlock.....	21
endDownload.....	22
APPENDIX A: QUERY CONDITION.....	24
Building a Condition.....	24
Condition Rules.....	24
Table of Supported Operators.....	24
Dates.....	25
APPENDIX B: CLIENT EXCEPTION CODES.....	26
APPENDIX C: SERVER EXCEPTION CODES.....	27

Introduction

The State API enables states to integrate their systems with SERFF application data. The goal of State API version 4.0 is to use Oracle as the persistence layer instead of Lotus Notes. The State API has usage limitations in order to regulate system load.

Overview

The State API web services can be grouped into six categories: Data Request Services, Correspondence Services, Update Filing Services, Aggregate PDF Services, Upload Attachment Handling Services, and Download Attachment Handling Services.

Data Request Services query for information and attachments on a filing. The web service is `queryFiling`.

Correspondence Services submit an Objection Letter, Note To Filer, or a Disposition. This grouping has six web services: `submitObjectionLetter`, `finalizeObjectionLetter`, `submitNoteToFiler`, `finalizeNoteToFiler`, `submitDisposition`, and `finalizeDisposition`.

Update Filing Services update three data values on a filing: Reviewer Name, State Tracking Number, and State Status. This grouping has three web services: `setReviewerName`, `setStateTrackingNumber`, and `setStateStatus`.

Aggregate PDF Services create a single Adobe Acrobat Portable Document Format (PDF) file of the entire filing and all compatible attachments. The web service is `Pdf_PdfRequest`.

Finalize for PDF Services destroys the temporary file created when a PDF is downloaded. The web service is `finalizePdf`.

Upload Attachment Handling Services and Download Attachment Handling Services manage the upload and download of file attachments to or from a SERFF Filing or Aggregate PDF. These services are designed to provide flexibility with large files.

Usage Limitations

The State API limits web service calls in order to regulate system load. These limits are imposed per minute and per hour.

When an initial web service call is made, two time stamps are stored on the system: one to provide a reference for calls per minute and one for calls per hour. Each call within a minute of the minute timestamp is counted against the maximum allowed web service calls per minute AND against the maximum allowed web service calls per hour. If either call limit is exceeded, a SOAP fault is returned containing two elements:

- A message describing the call limit exceeded
- The number of milliseconds until the next allowable call.

Once a call limit is exceeded, no further successful web service calls can be made until after the call limit expires. Once a call limit expires, a new time stamp for that call limit is not stored until a web service call is made.

Security and Authentication

Security is handled using HTTP authentication, a user name, and a password.

Shared Object Detail

Shared objects are used by more than one web service.

attachmentIdentifier

Contains two string values: *attachmentName* and *attachmentId*. These values are used by several web services in the SERFF API.

pendingReport

Contains one or more attachmentIdentifier shared objects and one string value: *temporaryReportId*, which is used by finalizeReport.

ClientException

A ClientException generally indicates an issue that the client needs to address in their application, such as missing data or invalid combinations of data.

A ClientException contains a *ClientException* object with two string elements:

- code – a predefined code indicating the error.
- message – a descriptive message of the error.

A table of exception codes can be found in Appendix B: Client Exception Codes.

ServerException

A ServerException generally indicates an issue with the web services application on the server.

A ServerException contains a *ServerException* object with two string elements:

- code – a predefined code indicating the error.
- message – a descriptive message of the error.

A table of exception codes can be found in Appendix C: Server Exception Codes.

CallRateExceededException

A CallRateExceededException indicates that the client has reached the web service Usage Limitation.

A CallRateExceededException contains a *CallRateExceededException* object with two string elements:

- millisecondsUntilNextAllowedCall – the number of milliseconds until the next allowed web service call.
- message – a descriptive message of the error.

Filing Services

QueryFiling

Input

A queryFiling object.

Returns

A maximum of 600 queryFilingResponse objects.

Conditions

The queryFiling object condition string value must be valid.

Results

Information is returned from SERFF. No changes to SERFF are made.

Object Detail

field

Contains one string value representing a field name whose value you want to return.

fields

Contains one or more field objects.

queryFiling

Contains one fields object and three string values:

- 1) *form*
Indicates the name of the object type you want returned (e.g., Filing).
- 2) *condition*
See Appendix A: Query Condition for information on writing a condition.
- 3) *attachmentNamePattern*
 - Leave the value blank (null), and no attachmentIdentifier's will be returned.
 - Use an asterisk (*), and an *attachmentIdentifier* will be returned for each attachment on the filing documents found by *condition*.
 - Use a string representing a partial or entire file name. An *attachmentIdentifier* will be returned for each attachment whose file name contains that string. Only

attachments on the filing documents found by *condition* will be evaluated.

queryFilingResponse

Contains one or more resultTableType objects.

resultTableType

Contains one or more row objects.

row

Contains one or more value objects and an array of attachmentIdentifier shared objects.

value

Contains one or more *vs* (value string) values. Each *vs* object represents one of potentially multiple values of a column in a row.

Correspondence Services

submitObjectionLetter

Submit an Objection Letter to SERFF.

Input

A submitObjectionLetterRequest object.

Returns

A submitObjectionLetterResponse object.

Conditions

- The *SerffTrackingNumber* string must reference a unique Filing.
- The target filing cannot have a closed status.

Results

- The Objection Letter is created on the Filing.
- When the first Objection Letter is created on a filing, the SERFF Status will change to “Pending Industry Response”.
- The SERFF log is updated when an Objection Letter is submitted.

Attachments

Attachments specified in the submitObjectionLetter object are not included in the Objection Letter until Upload Attachment Handling Services are called.

The Objection Letter will not be submitted to industry in SERFF until all specified attachments have been uploaded and finalizeReport is called.

Object Detail

submitObjectionLetter

Contains values as follows:

- filingId (string) (required)
- introductionText (string) (required)
- conclusionText (string)
- objectionLetterStatus (string) (required)
- effectiveDate (date) (required)
- attachments (string array)

ObjectionLetterResponse

- attachmentIdentifier

finalizeObjectionLetter

If any attachments are submitted on an Objection Letter, this service must be called at the end of the process.

Input

A finalizeObjectionLetterRequest object.

Returns

A finalizeObjectionLetterResponse object.

Conditions

- *objectionLetterId* must reference a pending submitObjectionLetter.
- All attachments must have been successfully uploaded using Upload Attachment Handling Services and completed using endUpload.

Results

- The Objection Letter indicated by *objectionLetterId* will be created and submitted in SERFF.
- If attachments are included, they will appear on the Objection Letter if successfully uploaded using Upload Attachment Handling Services.

Object Detail

finalizeObjectionLetter

Contains values as follows:

- objectionLetterId (string) (required)

finalizeObjectionLetterResponse

- Contains one string value representing the status of the Objection Letter: "SUCCESS".

UpdatefinalizeObjectionLetter

If any attachments are submitted on an Objection Letter, this service must be called at the end of the process.

Input

An UpdatefinalizeObjectionLetterRequest object.

Returns

An UpdatefinalizeObjectionLetterResponse object.

Conditions

- *objectionLetterId* must reference a pending submitObjectionLetter.
- All attachments must have been successfully uploaded using Upload Attachment Handling Services and completed using endUpload.

Results

- The Objection Letter indicated by *objectionLetterId* will be created and submitted in SERFF.
- If attachments are included, they will appear on the Objection Letter if successfully uploaded using Upload Attachment Handling Services.

Object Detail

finalizeObjectionLetter

Contains values as follows:

- objectionLetterId (string) (required)

finalizeObjectionLetterResponse

- Contains one string value representing the status of the Objection Letter: "SUCCESS".

submitNoteToFiler

Input

A submitNoteToFilerRequest object.

Returns

A `submitNoteToFilerResponse` object.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- The *SerffTrackingNumber* string must reference a unique Filing.

Results

- A Note to Filer is created on the Filing in SERFF.
- The SERFF Log document for the filing is updated.

Attachments

Attachments specified in the `submitNoteToFiler` object are not included in the Note to Filer until Upload Attachment Handling Services are called. The Note To Filer will not be created in SERFF until all specified attachments have been uploaded and `finalizeReport` is called.

Object Detail

submitNoteToFiler

Contains values as follows:

- `SerffTrackingNumber` (string) (required)
- `subject` (string) (required)
- `comments` (string) (required)
- `attachments` (string array)

NoteToFilerResponse

- `attachmentIdentifier`

finalizeNoteToFiler

If any attachments are submitted on a Note To Filer, this service must be called at the end of the process.

Input

A `finalizeNoteToFilerRequest` object.

Returns

A `finalizeNoteToFilerResponse` object.

Conditions

- *noteToFilerId* must reference a pending `submitNoteToFiler`.

Web Services

- All attachments must have been successfully uploaded using Upload Attachment Handling Services and completed using endUpload.

Results

- The Note To Filer indicated by *noteToFilerId* will be created and submitted in SERFF.
- If attachments are included, they will appear on the Note To Filer if successfully uploaded using Upload Attachment Handling Services.

Object Detail

finalizeNoteToFiler

Contains values as follows:

- *noteToFilerId* (string) (required)

FinalizeNoteToFilerResponse

- Contains one string value representing the status of the Note To Filer: "SUCCESS".

submitDisposition

Input

A *submitDispositionRequest* object.

Returns

A *submitDispositionResponse* object.

Conditions

A *ClientException* will be generated if the following conditions are not met:

- The *SerffTrackingNumber* string must reference a unique Filing.
- The target filing cannot have a closed status.
- The SERFF Status is updated when the Disposition is submitted.
- The SERFF log is updated when a Disposition is submitted.

Results

- A Disposition is created in SERFF as a response to the Filing indicated by *SerffTrackingNumber*.
- The SERFF Log for the filing is updated.

Attachments

Attachments specified in the `submitDispositionReport` object and are not included in the Disposition until Upload Attachment Handling Services are called. The Disposition will not be created in SERFF until all specified attachments have been uploaded and `finalizeReport` is called.

Object Detail

DispositionRequest

Contains values as follows:

- `filingId` (string) (required)
- `dispositionDate` (date) (required)
- `effectiveOrImplementationDate` (date)
- `effectiveRenewalDate` (date)
- `status` (string) (required)
- `comments` (string) (required)
- `sumOverallRateImpact` (number)
- `sumAffectedPolicyHolders` (number)
- `sumPremiumRateChange` (number)
- `attachments`
- Rate Change Information for each filing company when applicable:
 - `Cocode` (string) (required only if multiple companies)
 - `overallRateImpact` (number)
 - `premiumRateChange` (number)
 - `affectedPolicyHolders` (number)
 - `premium` (number)
 - `maxChange` (number)
 - `minChange` (number)

DispositionResponse

- `attachmentIdentifier`

finalizeDisposition

If any attachments are submitted on a Disposition, this service must be called at the end of the process.

Input

A `finalizeDispositionRequest` object.

Returns

A `finalizeDispositionResponse` object.

Conditions

- *DispositionId* must reference a pending submitDisposition.
- All attachments must have been successfully uploaded using Upload Attachment Handling Services and completed using endUpload.

Results

- The Disposition indicated by *DispositionId* will be created and submitted in SERFF.
- If attachments are included, they will appear on the Disposition if successfully uploaded using Upload Attachment Handling Services.

Object Detail

finalizeDisposition

Contains values as follows:

- dispositionId (string) (required)

finalizeDispositionResponse

- Contains one string value representing the status of the Objection Letter: "SUCCESS".

Update Filing Services

setReviewerName

Adds/changes the Reviewer name on a Filing in SERFF.

Input

A setReviewerNameRequest object.

Returns

A setReviewerNameResponse object.

Conditions

A ClientException will be generated if the following conditions are not met:

- The *SerffTrackingNumber* string must reference a unique Filing.
- The target filing cannot have a closed status.
- The Reviewer Name must be a valid user for that state instance.

Results

- The name of the State Reviewer on the Filing equals the input value for Reviewer Name.
- The SERFF status is set to “Assigned.”
- The SERFF Log document for the filing is updated.

Object Detail

setReviewerName

Contains values as follows:

- `serffTrackingNumber` (string) (required)
- `primaryReviewerName` (string) (required)
- `reviewerName` (string)

ReviewerNameResponse

- Contains one string value: SUCCESS.

setStateTrackingNumber

Adds/changes the State Tracking Number on a Filing in SERFF.

Input

A `setStateTrackingNumberRequest` object.

Returns

A `setStateTrackingNumberResponse` object.

Condition

A `ClientException` will be generated if the following conditions are not met:

- The *SerffTrackingNumber* string must reference a unique Filing.
- The target filing cannot have a closed status.

Results

- The State tracking number is set to the input value.
- The SERFF Log document for the filing is updated.

Object Detail

setStateTrackingNumber

Contains values as follows:

- `serffTrackingNumber` (string) (required)
- `stateTrackingNumber` (string) (required)

StateTrackingNumberResponse

- Contains one string value: SUCCESS.

setStateStatus

Change the State Status on the Filing in SERFF.

Input

A `setStateStatusRequest` object.

Returns

A `setStateStatusResponse` object.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- The *SerffTrackingNumber* string must reference a unique Filing.
- The target filing cannot have a closed status.

Results

- The State Status is set to the input value.
- The SERFF Log document for the filing is updated.

Object Detail

setStateStatus

Contains values as follows:

- `serffTrackingNumber` (string) (required)
- `stateStatus` (string) (required)

StateStatusResponse

- Contains one string value: "SUCCESS".

Aggregate PDF Services

RequestPdf

Creates a single Adobe Acrobat Portable Document Format (pdf) file of the entire filing and all compatible attachments.

Input

A `requestPdf` object.

Returns

A `PdfResponse` object.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- The *SerffTrackingNumber* string must reference a unique Filing.
- The target filing cannot have a draft status.
- Attachments must be in PDF format and cannot be larger than 3mb. If an attachment that is not in PDF format is included in the filing, the service will create a placeholder that includes the attachment name.

Results

- An `attachment_id` is returned, which can be used as a key to retrieve the result PDF file using Download Attachment Handling Services.

Object Detail

RequestPdf

Contains the value as follows:

- `serffTrackingNumber` (string) (required)

PdfResponse

- Contains `AttachmentIdentifier`.

FinalizePdf

This service is exempt from the Throttle.

Input

A `FinalizePdfRequest` object.

Returns

A `FinalizePdfResponse` object.

Conditions

- *AttachmentId* must reference an attachment.

Results

- The temporary file created when the PDF was downloaded is destroyed.

FinalizePdfRequest

Contains values as follows:

- `attachmentId` (string) (required)

FinalizePdfResponse

- Contains one string value representing the status: "SUCCESS".

Handling Attachments Services

This API sends and receives attachments. In order to handle large attachments, the client calls a series of services. The following three API functions allow a client to send an attachment:

beginUpload

This service must be called before uploading any file data using `writeBlock`. Subsequent calls before a call to `endUpload` have no effect.

Input

A `beginUploadRequest` object.

Returns

A `maxBlockSize` is returned, which should be used by `writeBlock`.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- *attachmentId* must reference a valid attachment id. *attachmentId* must be obtained through results of other web services in the SERFF State API. Attachments cannot be uploaded without first using one of the other web services in the SERFF State API to obtain an `attachmentId`.

Results

- The system creates a temporary file to hold the attachment.
- The system is prepared to receive files block-at-a-time through `writeBlock`.

Object Detail

`beginUploadRequest`

Contains values as follows:

- `attachmentId` (string) (required)
- `size` (integer) (required) (byte size of file)

`beginUploadResponse`

- Contains one integer value: `maxBlockSize`, representing the maximum block size, in bytes, that can be sent by `writeBlock`.

writeBlock

This service supports overwriting data so that the same block can be sent again should a failure occur. It will be available until `endUpload` is used on the attachment.

Input

A `writeBlockRequest` object.

Returns

A `writeBlockResponse` object.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- *attachmentId* must reference a valid attachment id.
- `beginUpload` must first have been called.
- The size of the base 64 decoded data must be less than the *maxBlockSize* obtained from `beginUploadResponse` in `beginUpload`.
- The sum of the file offset and the size of the decoded data must not be greater than the file size specified in `beginUpload`.

Results

- The system stores the data in the temporary file at the offset given by the input parameter.
- An *MD5 checksum* of the uploaded block is returned.

Object Detail

writeBlock

Contains values as follows:

- *attachmentId* (string) (required)
- *offset* (integer) (bytes) (required)
- *data* (hexBinary) (required)

writeBlockResponse

- Contains one string value, *md5*, indicating the MD5 checksum of the block uploaded.

endUpload

This service indicates that the upload for the attachment indicated by *attachmentId* is complete. Until `finalizeReport` is called, any `Upload Attachment Handling Services` may be called again since the report still is

Web Services

pending. However, once `endUpload` is called, individual blocks no longer are available so all blocks must be re-uploaded.

Input

An `endUploadRequest` object.

Returns

An `endUploadResponse` object.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- *attachmentId* must reference a valid attachment id.
- `writeBlock` must have been called at least once.

Results

- The attachment is created.
- An *MD5 checksum* of the file is returned.

Object Detail

`endUpload`

- `attachmentId` (string) (required)

`endUploadResponse`

- Contains one string value, *md5*, indicating the MD5 checksum of the attachment.

Download Attachment Handling Services

The Download Attachment Handling web services greatly facilitate downloads with large file attachments.

beginDownload

This service must be called before downloading any file data using `readBlock`. It enables you to download a temporary file one block-at-a-time, and to re-read portions that fail the *MD5 checksum*. Subsequent calls before calling `endDownload` have no effect.

Input

A `beginDownloadRequest` object.

Returns

A `beginDownloadResponse` object.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- *attachmentId* must reference a valid attachment id. *attachmentId* must be obtained through results of other web services in the SERFF State API. Attachments cannot be downloaded without first using one of the other web services in the SERFF State API.
- A *maxBlockSize* must be specified. This number must indicate the maximum block size that can be accepted for download. A value of zero indicates use of the server's default block size.

Results

- The requested resource is located and prepared for download.
- A *blockSize* is returned, which indicates the maximum block size that the server will send when using `readBlock`.
- A *fileSize* is returned, which indicates the total file size of the attachment to be downloaded.

Object Detail

`beginDownload`

- `attachmentId` (string) (required)
- `maxBlockSize` (integer) (required) (bytes)

`beginDownloadResponse`

Contains two integer values: *filesize* and *blockSize*, both in bytes.

readBlock

This service supports re-reading data so that the same block can be read again should a failure occur. It will be available until `endDownload` is used on the attachment.

Input

A `readBlockRequest` object.

Returns

A `readBlockResponse` object

Conditions

A `ClientException` will be generated if the following conditions are not met:

- *attachmentId* must reference a valid attachment id.

Web Services

- `BeginDownload` must first have been called.
- The size of the base 64 decoded data must be less than the *maxBlockSize* obtained from `beginDownloadResponse` in `beginDownload`.
- The sum of the file offset and the size of the decoded data must not be greater than the file size specified in `beginDownloadResponse` in `beginDownload`.

Object Detail

readBlock

Contains values as follows:

- `attachmentId` (string) (required)
- `offset` (integer) (required) (bytes)
- `blockSize` (integer)

readBlockResponse

Contains values as follows:

- `data` (hexBinary)
- `md5` (string)

endDownload

This service will clean up any temporary resources required for the download.

Input

An `endDownloadRequest` object.

Returns

An `endDownloadResponse` object.

Conditions

A `ClientException` will be generated if the following conditions are not met:

- *attachmentId* must reference a valid attachment id.
- `beginDownload` must have been called.

Results

- The system removes any temporary resources created by `beginDownload`.

Object Detail

endDownload

- `attachmentId` (string) (required)

Web Services

endDownloadResponse

- Contains one boolean value indicating success of the call.

Appendix A: Query Condition

Building a Condition

A condition is analogous to a single, simple SQL WHERE clause. A condition primarily uses SQL syntax.

Condition Rules

- Conditions do not support selects
- Conditions do not support joins
- Conditions do not support nesting
- Conditions do not support math expressions
- String literals must be enclosed in apostrophes (')
- String literals are case sensitive.
- If a string literal contains an apostrophe ('), it can be escaped by preceding it with another apostrophe ('). For example: 'Gilligan' 's Island'.
- If a LIKE or NOT LIKE clause contains a literal with one of the supported SQL wildcard characters (% or _), or the wildcard escape character (~), it must be escaped by preceding it with the wildcard escape character (~). For example: LIKE 'Average ~% Of Sales'. Without escaping these characters, they will be treated as wild cards and not literals.
- Multiple values can be separated with a comma (,).
- Date operators are not SQL-compatible and use a custom format.

Table of Supported Operators

=	TEXTFIELD = 'FieldValue' NUMBERFIELD = 4
<>	TEXTFIELD <> 'Not This FieldValue' NUMBERFIELD <> 3
<	NUMBERFIELD < 10
>	NUMBERFIELD > 2
<=	NUMBERFIELD <= 8
>=	NUMBERFIELD >= 3
% (for LIKE and NOT LIKE clauses only)	LIKE 'FieldV%e'
_ (for LIKE and NOT LIKE clauses only)	NOT LIKE 'FieldVal_e'
AND	TEXTFIELD = 'FieldValue' AND NUMBERFIELD <> 4
BETWEEN	NUMBERFIELD BETWEEN 3 AND 5

Appendix A: Query Condition

IN	TEXTFIELD IN (3, 4, 5)
LIKE	TEXTFIELD LIKE 'FieldV%e'
NOT BETWEEN	NUMBERFIELD NOT BETWEEN 2 AND 3
NOT IN	TEXTFIELD NOT IN (2, 9)
NOT LIKE	TEXTFIELD NOT LIKE 'FieldV_e'

Dates

Dates comprise a formatted string literal preceded by the DATE keyword. The string literal requires a date but a time component is optional. Formats are:

```
YYYY-MM-DD  
YYYY-MM-DD HH: MM: SS
```

For example, July 20, 1969 in the condition would be:

```
DATE '1969-07-20'
```

July 20, 1969 at 3:17:40 p.m. CDT would be:

```
DATE '1969-07-20 15: 17: 40'
```

NOTE: All times are in 24-hour format. Time zone cannot be specified, so all dates and times are compared without adjusting for time zone.

Appendix B: Client Exception Codes

Code	Description
1000	Undefined.
1001	Required fields missing.
1002	Invalid parameter.
1003	The query exceeded the maximum number of rows.
1004	No objects were found.
1005	The fields contain no data.
1006	Call rate limit exceeded.
1007	Parsing error.
1008	Invalid field for query.
1009	Fields to return must be provided.
1010	Filing not found for SERFF Tracking Number.
1011	Company represented by cocode not found for SERFF Tracking Number.
1012	Schedule Item represented by Reference ID not found for SERFF Tracking Number.
1013	Cannot create a Disposition or Objection Letter.
1014	State API User and Reviewer must be a STATE User Account Type.
1015	Unable to obtain the SAPI User or Instance not found for SAPI User or Reviewer.
1016	Unable to obtain the new Primary Reviewer Account.
1017	Reviewer Name does not exist.
1018	The Reviewer name provided is not associated with the filing instance.
1019	Correspondence item not found for id: _____.

Appendix C: Server Exception Codes

Code	Description
2000	Undefined.
2001	Internal database error.
2002	Internal application error.
2003	Missing resource.
2004	Malformed attachment id.
2005	Filing not found.
2006	General error encountered.
2007	IO error occurred.
2008	Client key is null.
2009	Null pointer exception.
2011	Remote PDF Conversion Failure.