Cleo VersaLex®

VersaLex API Guide

Version 5.4.1

June 2017

Cleo
4949 Harrison Avenue, Suite 200
Rockford, IL 61108 USA
Phone: +1.815.654.8110
Fax: +1.815.654.8294
Email: sales@cleo.com
www.cleo.com

Support: 1.815.282.7894, 1.866.444.2536 (US only), 02038653439 (UK), or support@cleo.com

Contents

Chapter

1

VersaLex API modes

Visit *Javadoc* for information about each available Java class and interface.

Client

API application process attaches to VersaLex server process over RMI; must have access to LexiCom/VLTrader/Harmony install directory. Use `LexiComFactory.getVersaLex(…, LexiComFactory.CLIENT_ONLY)` to vend.

Server

VersaLex server runs inside of API application process. Use `LexiComFactory.getVersaLex(…, LexiComFactory.SERVER_ONLY)` to vend.

Embedded

Available with the Cleo Harmony and Cleo VLTrader applications only.

- API implementation runs inside of the Cleo Harmony or Cleo VLTrader server process. Use `LexiComFactory.getCurrentInstance()` to vend. Copy the .jar file to `lib/api` and activate the class using one of the custom class properties in the Other Properties page in the Cleo Harmony, Cleo VLTrader and Cleo LexiCom applications.
- You can write URI schemes to access payload outside of the standard file system or database payload.
- The JavaScript option via the `SCRIPT` command also offers access to the VersaLex API using the ISessionScript interface.

Available with the Cleo Harmony application only.

- In addition to the `SCRIPT` command within actions, actions themselves can be fully JavaScript, either as host/mailbox actions or as standalone actions. And unlike host/mailbox actions, standalone actions can access other host/mailboxes and their events and payload.

Chapter

2

Possibilities

Trading partner configuration

Configure trading partner relationship, including certificate generation, export/import, and schedule. Associate external trading partner IDs.

- *ILexiCom*.{activateHost(), setProperty(), addID(), create(), clone(), save()}
- *ICertManagerRunTime*
- *CertificateInfo*
- *ISchedule*

Sending/receiving files/streams

Through trading partner mailboxes/actions using:

- Higher-level
  - *IMailboxController* – uses default mailbox <send> and <receive> actions and their commands to send and receive via temporary actions (which allows concurrency). VersaLex router, DB payload, integration web service, and resend features make use of IMailboxController.
  - *IActionController* – execute action commands (*LexCommand*) one-by-one. May or may not be a temporary action.
  - *IListenerController* – for reacting to and influencing receives only.
- Lower-level
  - *LexiComOutgoing* – outgoing file or stream wrapper
  - *ILexiComIncoming* – incoming file or stream interface
  - *ILexiComMultipartIncoming* – incoming multipart message file or stream interface
  - *LexURIFile* – Customer-written or Cleo-provided URI scheme(s) used to access payload. URI schemes may be used for the host Inbox and Outbox and as the source or destination of action commands.

System log listener

LexiComLogListener – watch all system XML log events to process newly received files (similar to ExecuteOnSuccessfulReceive advanced property), to send an alert on failure (similar to EmailOnFail advanced property), to monitor traffic…

System options

Options – default in/out directories, database connection parameters, branding, misc. properties…

High-availability synchronization status

ISyncer, *Sync* – for checking if this VersaLex is currently active or passive and, if active, whether it is the master VersaLex.

Recovering from server restart

ILexiCom.noop() – for detecting if VersaLex server has been restarted.

Licensing and registration

ILicense, *ILicenser*, *SystemInfo*, *RegistrationInfo* – for automating registration and licensing so not necessary by operator.

EDI parser

EDI, *EDIFilterInputStream*, *EDISegment*, *EDIElement* – parses X12, EDIFACT, TRADACOMS.

Utility functions

LexUtil – send email (non-payload), db connection pool, get debug writer.

Chapter

3

Java API Examples

This section contains examples from `src/examples/`.

Configure trading partner mailboxes

`ConfigureAS2HostMailbox.java`

`ConfigureFTPHostMailbox.java`

`ConfigureOFTPHostMailbox.java`

Configure AS2 service

`ConfigureAS2Listener.java`

Configure trading partner mailbox and then send files via IMailboxController

`CreateAndUseFTPHost.java`

`CreateAndUseSSHFTPHost.java`

IActionController example

`ActionController.java`

Embedded incoming classes

`EmbeddedILexiComIncoming.java`

`EmbeddedFilterOutputStream.java`

Embedded outgoing class

`EmbeddedLexiComOutgoingThread.java`

Example log listeners

`EmbeddedLexiComLogListener.java`

`ServerOrClientLogListener.java`

Example showing VersaLex as a streaming integration proxy

`EmbeddedSMGILexiComIncoming.java`

For client mode, demonstrates how to test for VersaLex restart and reconnect

`TestAPI.java`

Automated product registration and licensing example

`RegisterProduct.java`

`AcquirePermanentLicense.java`

`UnregisterProduct.java`

Chapter

4

Standalone JavaScript action examples

Note: For both the incoming file filter and log listener, the expectation is that the incoming and log functions return in a timely fashion.

Incoming File Filter Template

```
importPackage(com.cleo.lexicom.external);
incomingfilter();
/*
* Filter in-scope incoming files
*/
function incomingfilter() {
  ISessionScript.getIncomingFilters().add(new LexiComIncomingFilter() {
   incoming: function(mailbox, file, map) {
     // do some prep work for the incoming file and/or return an output
 stream to be used
     return null;
    }
  });
}
```

Log Listener Template

```
importPackage(com.cleo.lexicom.external);
loglistener();
/*
* Listen for in-scope log events
*/
function loglistener() {
  ISessionScript.getLogListeners().add(new LexiComLogListener() {
    log: function(event) {
      // do something with the log event
    }
  });
}
```