Automic
Let's Automate Business.

February 2015

Automic JAVA API

Getting Started

Brendan Sapience – bsp@automic.com

Solution Architect

Our Goal

Automic
Let's Automate Business.

- Understanding how the AE Java API works

Prerequisites

Automic
Let's Automate Business.

- You need some Java exposure (familiarity with Java is recommended)
- A Java IDE must be installed (Eclipse is recommended for this class)
- An AE instance must be available for testing (Dedicated Client Recommended)

Automic
Let's Automate Business.

Setting up the Environment

How do I create my project and where is the AE Java API?

Setting up a Standard Java project

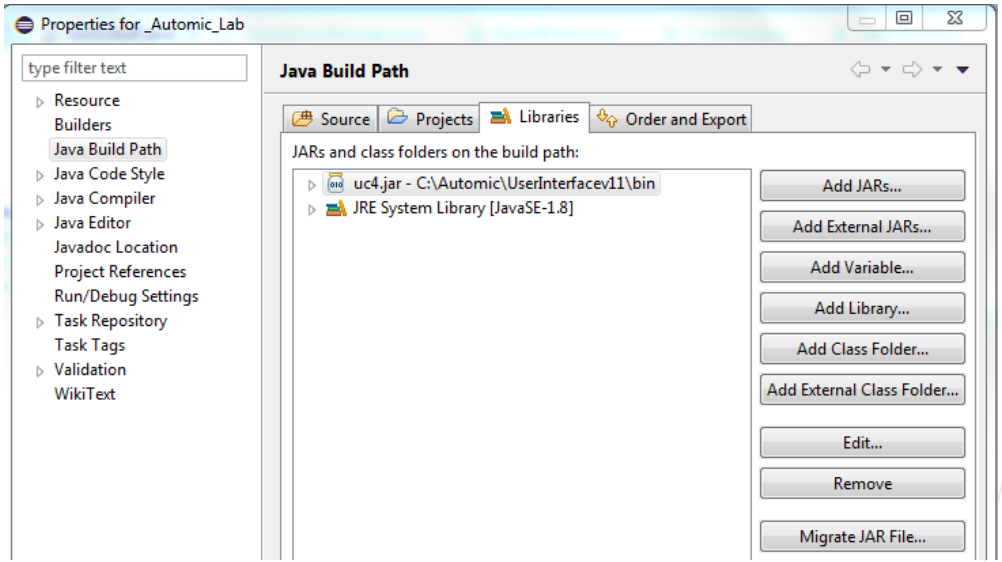
Automic
Let's Automate Business.

- Open your Java IDE (ex: Eclipse)
- Create a new Java project
- Add one class
- Add a **main** method to the class
- Display a simple message in the main method:

```
public static void main(String[] args) {
        System.out.println("Hello Automic");
}
```

Adding the AE Java API

- Locate the **uc4.jar** file from the **bin directory** of your **UserInterface** installation (*ex: c:\automic\userInterface\bin\uc4.jar*)
- In your Java project, edit the **Java Build Path** (via the IDE)
- In the Java Build Path, add **uc4.jar** as an "**external Library**":



Automic
Let's Automate Business.

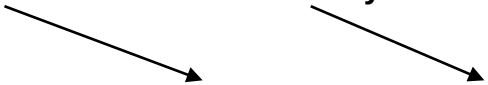
Connecting to the Automation Engine

How do we first authenticate to AE in Java?

Connecting to AE

The **Connection Object** is central to the API use and should be the first one to be used:

- The **open** method requires the **AE IP Address** and the **AE Primary CP Port**



```
Connection myConnection = Connection.open(192.168.126.12, 2217);

CreateSession mySession = myConnection.login(0,”BSP”, “DEPT”, “Un1ver$e”, ‘E’);

if(mySession.getMessageBox()!=null){
System.out.println("-- Error: " + mySession.getMessageBox()); return null;
}
```

The **login** method from the Connection object handles **authentication to the AE**, it requires :

- Client Number (between 0 and 9999, as an Integer)
- Login (as a String)
- Department (as a String)
- Password (as a String)
- Language (as a character)

Catching Error Messages

Errors Messages are generally stored as a **MessageBox** Object. It provides detailed information on the error, such as:

- Access Denied
- AE impossible to reach

```
Connection myConnection = Connection.open(192.168.126.12, 2217);

CreateSession mySession = myConnection.login(0,”BSP”, “DEPT”, “Un1ver$e”, ‘E’);

if(mySession.getMessageBox()!=null){
System.out.println("-- Error: " + mySession.getMessageBox()); return null;
}
```

The rule about MessageBoxes is simple:

- If it is null: **there was no error**, and the request was successful
- If it is not null: **something went wrong** and more information is provided within it

Exercise

Automic
Let's Automate Business.

1- Connect to your own AE using a Connection object, the open and login methods

2- Display the welcome message from your AE

3- Close the Connection

Automic
Let's Automate Business.

Interacting with AE via Requests

How do we ask AE to do stuff and get information back?

Sending Requests to AE

Automic
Let's Automate Business.

Doing stuff with the Java API is basically **sending various requests to AE** and waiting for a response.

The Connection object has a very important method to handle just that: **sendRequestAndWait**:

```
public void saveObject(UC4Object obj) throws IOException {
SaveObject save = new SaveObject(obj);
connection.sendRequestAndWait(save);
if (save.getMessageBox() != null) {
System.out.println(" -- "+save.getMessageBox().getText());
}
}
```

The **sendRequestAndWait** method takes a **Request** as a parameter.

The Request needs to be instantiated before **sendRequestAndWait** is used

- Above, the Request is a **SaveObject**.. Which is a way to save an object of any type (JOBS, JOBP etc.)
- The Request is instantiated with **Java Objects** that depend on the type of request

What Requests are available?

Automic
Let's Automate Business.

Requests are Java Classes from within the **uc4.jar** file(under **com.uc4.communication.requests**). They can all be found here:

http://docs.automic.com/documentation/AE/10.0.4/english/AE_API/pages/index.html

Here is an extract from the list of Requests available:

- CancelTask
- CreateObject
- **DuplicateObject**
- ForecastList
- IgnoreConditions
- ImportObject
- LatestReport
- ModifyStartTime
- OpenObject
- RestartTask
- **UserList**
- … (140+)

com.uc4.communication.requests

Class UserList

java.lang.Object
com.uc4.communication.requests.XMLRequest
com.uc4.communication.requests.UserList

All Implemented Interfaces:

java.lang.Iterable<UserListItem>

```
public class UserList
extends XMLRequest
implements java.lang.Iterable<UserListItem>
```

Lists all users in a client. All the users of the UC4 system may be viewed in system client "0000".

Constructor Summary

Constructors

Constructor and Description
UserList()

com.uc4.communication.requests

Class DuplicateObject

java.lang.Object
com.uc4.communication.requests.XMLRequest
com.uc4.communication.requests.DuplicateObject

```
public class DuplicateObject
extends XMLRequest
```

Duplicates an UC4 Object

Constructor Summary

Constructors

Constructor and Description
DuplicateObject(UC4ObjectName name, UC4ObjectName newName, IFolder folder) Creates a new DuplicateObject request.
DuplicateObject(UC4Object object, UC4ObjectName newName, IFolder folder) Creates a new DuplicateObject request.

3 Main Types of Requests

Automic
Let's Automate Business.

Generic Object Requests:

- Open Object
- Close Object
- Save Object
- Delete Object
- Create Object
- Duplicate Object
- Execute Object
- Export Object
- Import Object
- Move Object
- Rename Object
- Replace Object
- Search Object
- Transport Object

Common to all AE Objects (JOBS, JOBP, CALE etc.)

Object List Requests:

- Activity List
- Agent Group list
- Agent List
- Calendar list
- Client List
- Folder list
- Forecast List
- Queue List
- Server List
- Template List
- User List

Specific to Object Type

Specific Object Requests:

- RecalculateAutoForecast
- ReleaseManually
- RerunWorkflow
- …

Specific Actions

The Most Important Requests

Automic
Let's Automate Business.

Generic Object Requests:

- Open Object
- Close Object
- Save Object
- Delete Object
- Create Object
- Duplicate Object
- Execute Object
- Export Object
- Import Object
- Move Object
- Rename Object
- Replace Object
- Search Object
- Transport Object

Generally take at least a **UC4Object** as a parameter

They work for **All AE Objects** (depending on the context..)

Ex:

- Delete an existing Job
- Duplicate a Calendar
- Export Logins
- Search Workflows starting with ABC*

The Most Important Requests

Automic
Let's Automate Business.

Object List Requests:

- Activity List
- Agent Group list
- Agent List
- Calendar list
- Client List
- Folder list
- Forecast List
- Queue List
- Server List
- Template List
- User List

Generally take **no parameter at all** (notable exception: **Folder list**)

They always return a **List of Item** for further processing.

Ex:

- FolderList returns a list of **FolderListItems**
- AgentList returns a list of **AgentListItems**
- QueueList returns a list of **QueueListItems**

The Most Important Requests

Automic
Let's Automate Business.

Specific Object Requests:

- RecalculateAutoForecast
- ReleaseManually
- RerunWorkflow
- …

No Specific rule.. It behaves according to what it does.

Ex:

- RerunWorkflow returns nothing
- ReleaseManually returns nothing

Exercise

Automic
Let's Automate Business.

1- Connect to your own AE

2- Use the **AgentList** request, submit it to AE (hint: use the **sendRequestAndWait** method)

3- Display the **number of Agents** in AE from the Request

4- **Bonus Question**: display the name of each agent

Automic
Let's Automate Business.

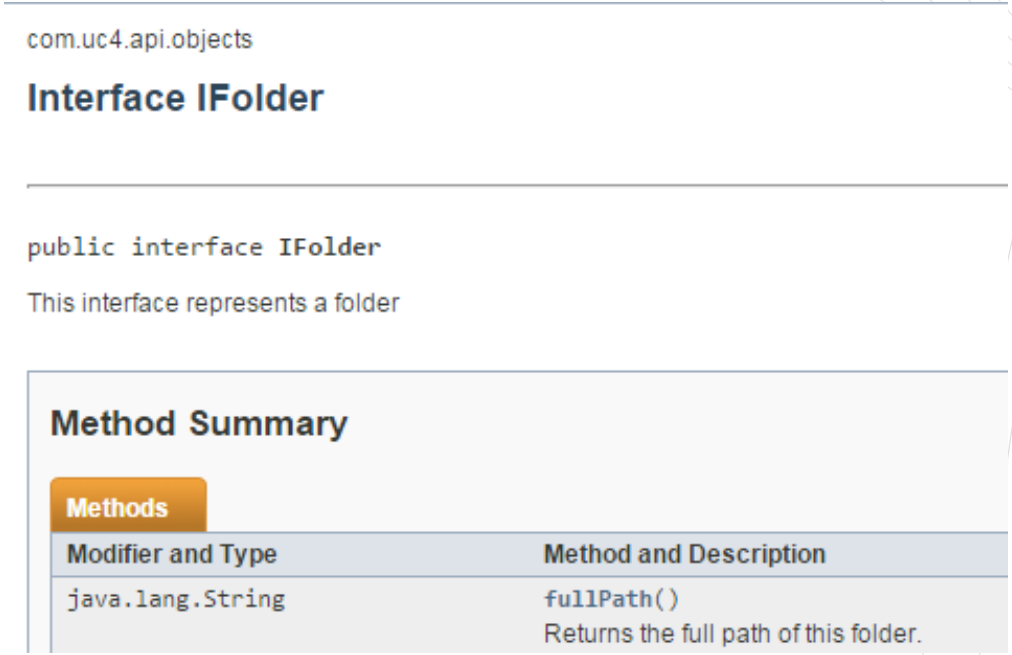
Working with Folders

How do we start exploring an existing AE Client in Java?

Working with Folders

A **Folder Object** in the API is called an **Ifolder:**

…



com.uc4.api.objects

Interface IFolder

```
public interface IFolder
```

This interface represents a folder

Method Summary

Methods

Modifier and Type	Method and Description
java.lang.String	fullPath() Returns the full path of this folder.

Working with Folders

The **FolderList** request **DOES NOT** return a **list of Folders**…

It returns a **List of Items** (FolderItems) within a given Folder (**IFolder**)!

…

com.uc4.communication.requests

Class FolderList

java.lang.Object
  com.uc4.communication.requests.XMLRequest
    com.uc4.communication.requests.FolderList

All Implemented Interfaces:

java.lang.Iterable<FolderListItem>

```
public class FolderList
extends XMLRequest
implements java.lang.Iterable<FolderListItem>
```

Lists the content of a folder.

Constructor Summary

Constructors

Constructor and Description
`FolderList(IFolder folder)` Constructs a new `FolderList` to list the content of the specified folder.
`FolderList(IFolder folder, boolean executableOnly)` Constructs a new `FolderList` to list the content of the specified folder.
`FolderList(IFolder folder, java.util.List<java.lang.String> objectTypes)`

Working with Folders

Automic
Let's Automate Business.

Getting to the root of them all: If I need an Ifolder object to retrieve its content.. **How do I get an Ifolder Object in the first place**?

Answer: **Using the FolderTree request (it returns the root IFolder)**:

```
public IFolder getRootFolder() throws IOException{
            FolderTree tree = new FolderTree();
            this.connection.sendRequestAndWait(tree);
            return tree.root();
}

// Returns a list of ALL Folders (including folders in folders, folders in folders in folders etc.)
public ArrayList<IFolder> getAllFolders(boolean OnlyExtractFolderObjects) throws IOException{
            ArrayList<IFolder> FolderList = new ArrayList<IFolder>();
            if(!OnlyExtractFolderObjects){FolderList.add(getRootFolder());}
            IFolder rootFolder = getRootFolder();
            Iterator<IFolder> it = rootFolder.subfolder();
            while (it.hasNext()){
                        IFolder myFolder = it.next();
                        if(! myFolder.getName().equals("<No Folder>")){
                                    addFoldersToList(FolderList,myFolder,OnlyExtractFolderObjects);
                        }
            }
            return FolderList;
}
```

Working with Folders

Automic
Let's Automate Business.

Getting to the root of them all: Now that we have the root folder.. How do we get **anywhere else**?

Answer: **Iterating** on the **content of the Root Folder** via method **subfolder**:

```
public IFolder getRootFolder() throws IOException{
            FolderTree tree = new FolderTree();
            this.connection.sendRequestAndWait(tree);
            return tree.root();
}

// Returns a list of ALL Folders (including folders in folders, folders in folders in folders etc.)
public ArrayList<IFolder> getAllFolders(boolean OnlyExtractFolderObjects) throws IOException{
            ArrayList<IFolder> FolderList = new ArrayList<IFolder>();
            if(!OnlyExtractFolderObjects){FolderList.add(getRootFolder());}
            IFolder rootFolder = getRootFolder();
            Iterator<IFolder> it = rootFolder.subfolder();
            while (it.hasNext()){
                        IFolder myFolder = it.next();
                        if(! myFolder.getName().equals("<No Folder>")){
                                    addFoldersToList(FolderList,myFolder,OnlyExtractFolderObjects);
                        }
            }
            return FolderList;
}
```

Exercise

Automic
Let's Automate Business.

1- Connect to your own AE

2- Use the **FolderTree** request to retrieve the **Root Folder** (IFolder object)

3- Use the **FolderList** request with the Root Folder object retrieved above

4- Display the **name** of **each item** contained in the **Root Folder** (Hint: the **FolderList** request will return an array of **FolderListItems**)

5- **Bonus Question**: find a way to get all IFolder for ALL folders in your client

6- **Ultra Bonus Question**: write a method that takes a folder name as a String parameter and returns the corresponding IFolder object

Automic
Let's Automate Business.

Working with Generic Objects and AE Objects

How do we modify / create / update AE Objects in Java?

Working with Specific Requests

Automic
Let's Automate Business.

Working Example: How do I suspend and resume on of my AE Clients?

Answer: By using the exact same request mechanism as the rest..

```
public void suspendClient() throws IOException{

OpenObject reqOpen = OpenObject(ClientName,false,true);
connection.sendRequestAndWait(reqOpen);
Client client = (Client) reqOpen.getUC4Object();
SuspendClient req = new SuspendClient();
connection.sendRequestAndWait(req);

}
```

Working with Generic Objects

Automic
Let's Automate Business.

Generic Object Requests are… **Generic**: They **ALWAYS** make use of **UC4Object** objects..

What is a UC4Object object?

Answer: **The superclass of all AE Objects**

com.uc4.api.objects

Class UC4Object

java.lang.Object
    com.uc4.api.objects.UC4Object

Direct Known Subclasses:

Calendar, Client, ConsoleEvent, Dashboard, DatabaseConnection, DatabaseEvent, Documentation, FileEvent, FileTransfer, Group, Host, HostGroup, Include, Job, JobPlan, Login, Notification, OutputFilter, PromptSet, Queue, RAConnection, SAPConnection, SAPQueueManager, Schedule, Script, Sync, TimeEvent, TimeZone, User, UserGroup, Variable, WorkflowIF, WorkflowLoop

This is equivalent to saying that all **AE Objects** extends (inheritance in the Java sense) from **UC4Object**:

com.uc4.api.objects

Class Job

java.lang.Object
    com.uc4.api.objects.UC4Object
        com.uc4.api.objects.Job

```
public class Job
extends UC4Object
```

This is the base class for all jobs.

Working with Generic Objects

Automic
Let's Automate Business.

Working Example: **Opening an Object** (**OpenObject** request):

- **Input parameters**: String (Object Name), Boolean (read only), Boolean (full object)
- **returns**: UC4Object object

```
public UC4Object openObject(String name, boolean readOnly) throws IOException {

//Valid for Workflow, Jobs, Calendars etc.
UC4ObjectName objName = new UC4ObjectName(name);

// last boolean is for a full object.. Always set to true
OpenObject open = new OpenObject(objName,readOnly,true);
connection.sendRequestAndWait(open);

if (open.getMessageBox() != null) {
     System.err.println(" -- "+open.getMessageBox().getText());
     System.out.println(" %% Object: "+ name +" returned a message box: "+open.getMessageBox().getNumber());
}

// the request now contains a UC4Object object
return open.getUC4Object();
}
```

Working with Generic Objects

Automic
Let's Automate Business.

Now, how do we get **from a Generic Object** (**UC4Object**) to an **Actual Object** (ex: a **JOB**)?

Answer: By casting the **UC4Object** to whatever Object type it is.

Object Types available are: Calendar, Client, ConsoleEvent, FileEvent, FileTransfer, Job, JobPlan, Login, Notification, Queue etc.

You can see the object type with getType() method on the UC4Object object

```
UC4Object object = broker.common.openObject(jobName, true);
System.out.println(" UC4Object Type "+object.getType());
Job myJob = (Job) object;

//Modifications in the Job would occur here.

SaveObject save = new SaveObject(object);
connection.sendRequestAndWait(save);
if (save.getMessageBox() != null) {System.out.println(" -- "+save.getMessageBox().getText());}

CloseObject close = new CloseObject(object);
connection.sendRequestAndWait(close);
if (close.getMessageBox() != null) {System.err.println(" -- "+close.getMessageBox().getText());}
```

Don't Forget to Save and Close the Object!

Exercise

Automic
Let's Automate Business.

1- Connect to your own AE

2- Use the **OpenObject** request with one of your existing AE Job Names

3- Retrieve the **UC4Object** corresponding to your job from the **OpenObject** request

4- Retrieve the **Job** object from the **UC4Object** (**Hint**: use Java's casting mechanism)

5- Change the **Archive Key 1** of your Job to your first and last name

6- **Save** the Job object

7- **Close** the Job object

8- **Check** that the modification took place using the **User Interface**

9- **Bonus Question**: Build a method that takes a String as input (any Job name) and returns the corresponding Job object

Going Further

Automic
Let's Automate Business.

Check out the content of the following Packages:

com.uc4.api.objects

com.uc4.api.communication.requests

UC4.ApplicationInterface

Packages

Package	Description
com.uc4.api	This package contains classes which are generally used.
com.uc4.api.objects	This package contains classes related to UC4 objects.
com.uc4.api.prompt	This package contains prompt element classes.
com.uc4.api.systemoverview	This package contains classes to get information from the UC4 system overview.
com.uc4.communication	This package contains classes for the communication with the UC4 Server.
com.uc4.communication.requests	This package contains request classes that can be sent using the `Connection` class.

Clone (and Fork) the existing github project:

https://github.com/brendanSapience/UC4-Automic---Java-API-Framework-Simplified

branch: master ▾ UC4-Automic---Java-API-Framework-Simplified / src / com / automic / +

adding methods for Users, Usergroups and Logins

brendanSapience authored 7 hours ago — latest commit ffe6d18e47

..

factories	adding comments on factories	2 months ago
objects	adding methods for Users, Usergroups and Logins	7 hours ago
tests	sanitizing the code	10 hours ago
utils	adding classes for following objects: Links, Forecasts, executions,	4 months ago
AECredentials.java	porting codebase to github	4 months ago
ConnectionManager.java	better performance on object listing (no need to open each object..)	4 months ago
GoAutomic.java	sanitizing the code	10 hours ago
SupportedAEVersions.java	porting codebase to github	4 months ago