**[Welcome to the CIP4 Community Website](http://community.cip4.org/index.php/26-content/home/104-welcome-to-the-cip4-community-website)**

* 
* 

This Website aims on providing you tools and documentation to integrate JDF into your Workflow. Next to that it will serve as a platform for everyone to share thoghts, solve problems and to get in contact with experts. We want to encourage you to share your knowledge about JDF integration and help others in our Forums and to post your contact and skills into our expert-section to simplify networking.

[**Alces**](http://community.cip4.org/index.php/tools/alces)

* 
* 

Description

CIP4 [Alces](http://jdf4you.org/glossary/5#term48) is an JDF / JMF Integration Tool. Using [Alces](http://jdf4you.org/glossary/5#term48) you are able to explore JDF / JMF Interfaces. Further it is an easy tool for learning how JDF / JMF works. [Alces](http://jdf4you.org/glossary/5#term48) provides and showcases the mechanism how to send [JDF Document](http://jdf4you.org/glossary/5#term20)s to a target JDF Device using Job Messaging Format (JMF). The [JDF Document](http://jdf4you.org/glossary/5#term20) creation and modification is NOT covered by CIP4 [Alces](http://jdf4you.org/glossary/5#term48).

Documentation

On Windows Operating Systems CIP4 Alces can be launched by executing the batch file "alces.bat". For all other Operating Systems there also is a shell script "alces.sh". Both scipts initially start the Java Virtual Machine in a command line. When Alces is started successfully the blank Main Dialog (screenshot above) appears.

**Connect to JDF Device**



In order to connect to a JDF device you will need the URL of it. Best for testing is to set up Bambi on your local machine and use its virtual devices. If you haven't set it up so far, you can download it here and follow these instructions.

In order to connect to the JDF Device type in the URL in the text field "Device / Controller URL" located on top of Alces' Main Dialog. After pressing the button "Connect" Alces will try to connect the JDF Device. In the background a JMF Query Message "KnownMessages" (see JDF Spec. 1.4a - "5.8.5 KnownMessages") will be sent to the target Device. The synchronous JDF Response Message "ResponseKnownMessages" contains a list of all messages supported by the JDF Device.

After that, Alces will send each a "KnownDevices" and a "KnownSubscriptions" message and receive the response form the device. Finally all details will be shown in the main dialog.

In the panel "Known Messages" (on the left) buttons will appear for each known message. More details about the target JDF Device (or JDF Controller) are provided in the panel "Known Devices".

**JMF Messaging**

When connected you are able to run further JMF transactions. For job management each JDF Device generally provides a queue. So first of all lets do some queue operations:
The lower right panel in Main Dialog displays queue details of the JDF Device connected. Among other things there is a value for "Queue Status". Using JMF Command Messages the queue status can be changed. Global queue handling JMF Messages are specified in JDF Specification 1.4a "5.13 Messages for Global Handling of Queues".

All JDF Command Messages supported by the target device are shown as buttons in the "Known Messages" panel. When connected to a Bambi JDF Device among others there are buttons "OpenQueue", "HoldQueue", "CloseQueue" and "ResumeQueue". A click of one of these buttons sends a specific JMF Command Message to the JDF Device connected which changes the queue status.

**JMF Message History and Analysis**

The panel in the middle of the Main Dialog is called "JMF Message History and Analysis"-panel. Each JMF Message sent or received is listed in this panel. All messages are separated in transaction groups. Each group is displayed by a root node. The subnode of such a group represents the root message which initiated the transaction.

For example when you send a JMF Query Message "Status" to the target device a JMF Response Message containing the current device status is expected. The request and the response belong to one and the same transaction. So in the "JMF Message History and Analysis"-panel the response message will be shown as a subnode of the JMF Message which called it. In order to get more details about the XML content of a JMF Message  click on the item in center panel. The content is shown in the upper right content panel. If you want to learn how JMF works here you can track all messages and their effects.

**Transmission of JDF Documents**

The button "SubmitQueueEntry..." gives you the possibility to submit a JDF Document. On click a new dialog appears where you can select the JDF Document you want to submit. After selecting the document, a JMF command "SubmitQueueEntry" will be created and sent to the target device. The response message contains details whether or not the job transmission was successful. The whole message transaction is displayed in the "JMF Message History and Analysis" panel.

[**Bambi**](http://community.cip4.org/index.php/tools/bambi)

* 
* 

Description

Bambi is a Tomcat application, which simulates different JDF devices in a virtual print shop.

Documentation

Out of the box Bambi2 simulates a set of six common JDF Devices (Plate Making, Press and Guillotine). Configuration files can be found in [BAMBI\_WEB\_ROOT]/config. When Bambi is started for the first time all settings will be dumped to the preconfigured BaseDir (default: "/var/www/bambi-2/conf").

**Please care:** In order to modify settings please consider removing the dumped files and restart Bambi2!

### JDF Devices out of the box

|  |  |  |
| --- | --- | --- |
| **Device Id** | **Simulated Device** | **Simulated process step** |
| 01-Suprasetter105-1 | Suprasetter 105 No. 1 | Exposition device No. 1 |
| 02-Suprasetter105-2 | Suprasetter 105 No. 2 | Exposition device No. 2 |
| 03-SpeedmasterXL105-1 | Speedmaster XL 105-8-P No. 1 | Press No. 1 |
| 04-SpeedmasterXL105-2 | Speedmaster XL 105-8-P No. 2 | Press No. 2 |
| 05-Polar115-1 | Polar 115 No. 1 | Guillotine No. 1 |
| 06-Polar115-2 | Polar 115 No. 2 | Guillotine No. 2 |

### Base Configuration: devices.xml

The file devices.xml is the main configuration file of Bambi2. In this file you can modify the general settings (i.e. "BaseDir") as well as device specific settings.

Each type of device has a particular set of job phases. All job-phases for a device are configured in a separate file called "job\_[DEVICE\_ID].xml".

[?](http://community.cip4.org/index.php/tools/bambi)

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223242526272829303132333435363738394041424344454647 | <application BaseDir="/var/tmp/bambi-2/conf/" BaseUrl="bambi-2" JDFDir="JDFDir"     SenderID="Bambi2" Port="8080" CallBackName="org.cip4.bambi.extensions.ExtensionCallback">     <Device DeviceID="Root" DeviceType="Bambi 2 Root" />                <!-- Plate Making -->     <Device DeviceID="01-Suprasetter105-1" DeviceType="Suprasetter 105 Nr. 1"         TrackResource="Component:Output" InputHF="/var/tmp/bambi-2/Suprasetter105-1/Input"         OutputHF="/var/tmp/bambi-2/Suprasetter105-1/Output" ErrorHF="/var/tmp/bambi-2/Suprasetter105-1/Error" />     <Device DeviceID="02-Suprasetter105-2" DeviceType="Suprasetter 105 Nr. 2"         TrackResource="Component:Output" />        <!-- Press -->     <Device DeviceID="03-SpeedmasterXL105-1" DeviceType="Speedmaster XL 105-8-P Nr. 1"         TrackResource="Component:Output" InputHF="/var/tmp/bambi-2/SpeedmasterXL105-1/Input"         OutputHF="/var/tmp/bambi-2/SpeedmasterXL105-1/Output" ErrorHF="/var/tmp/bambi-2/SpeedmasterXL105-1/Error" />     <Device DeviceID="04-SpeedmasterXL105-2" DeviceType="Speedmaster XL 105-8-P Nr. 2"         TrackResource="Component:Output" />                 <!-- Guillotine -->     <Device DeviceID="05-Polar115-1" DeviceType="Polar 115 Nr. 1"         TrackResource="Component:Output" />     <Device DeviceID="06-Polar115-2" DeviceType="Polar 115 Nr. 2"         TrackResource="Component:Output" /> </application> |

### Most significant XML-Attributes

**application/@BaseDir**

Path where settings files will be dumped.

**application/@SenderID**

SenderID shown in JDF / [JMF Message](http://jdf4you.org/glossary/5#term21)s.

**application/@Port**

Just for display in all Device Urls

**application/Device/@DeviceID**

Unique device identifier

**application/Device/@InputHF**

Input Hotfolder to submit jobs.

**application/Device/@OutputHF**

Output folder for processed jobs.

**application/Device/@ErrorHF**

Output folder for failed jobs.

**application/Device/@TrackResource**

Tracked resource type.

### Device Job Phases

Each JDF Device has a particular set of Job Phases. A Job Phase is a production mode while processing a job. For example the press in Bambi consists of three Job Phases ("Setup", "Running" and Cleanup").

All except one of the JobPhase attributes are based on the JDF Specification. The attribute "Duration" simply defines the time in seconds, how long the simulation of a Job Phase does take. Real JDF-devices of course usually don’t have a defined time to run before they stop or initiate the next job phase, but run until the specific job phase is finished. In the simulation a JobPhase is finished when duration time is off or all resources are processed (defined by Speed).

[?](http://community.cip4.org/index.php/tools/bambi)

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223242526272829303132333435363738394041 | <?xml version="1.0" encoding="UTF-8"?> <BambiJob Error="3">     <JobPhase Duration="10"         DeviceStatus="Setup" DeviceStatusDetails="Waste"         NodeStatus="Setup" NodeStatusDetails="node setup">         <Amount Resource="Component:Output" Speed="4000" Waste="true"/>         <Amount Resource="Media:Input" Speed="4000" Waste="true" />         </JobPhase>     <JobPhase Duration="120"         DeviceStatus="Running" DeviceStatusDetails="Good"         NodeStatus="InProgress" NodeStatusDetails="producing node">         <Amount Resource="Component:Output" Speed="16000" />         <Amount Resource="Media:Input" Speed="16000"  />         </JobPhase>     <JobPhase Duration="2"         DeviceStatus="Cleanup" DeviceStatusDetails="WashUp"         NodeStatus="Cleanup" NodeStatusDetails="cleaning up">         <Amount Resource="Component:Output" Speed="0" />         <Amount Resource="Media:Input" Speed="0" />     </JobPhase> </BambiJob> |
|   |   |

Persons Check-in

### Status Subscriptions

To Start a StatusSubscription with Alces, connect to Bambi using [serverIP:Port/bambi-home]/jmf. Use the “Send File” button and submit a JMF like the Example below.

Enter the IP and port Number you want to receive the subscription messages with in the URL-attribute of the Subscription Element. The RepeatTime-attribute submits the frequency of status messages in seconds.

Before submitting the subscription remember to edit the Query ID to a value you haven’t used before. In this case you will get a subscription for each submission and Alces will display them correctly.

[?](http://community.cip4.org/index.php/tools/bambi)

|  |  |
| --- | --- |
| 1234567891011121314151617 | <?xml version="1.0" encoding="UTF-8"?> <JMF xmlns="<http://www.CIP4.org/JDFSchema_1_1>" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"  SenderID="Alces 0.9.9.1" TimeStamp="2013-02-12T16:01:08+01:00" Version="1.3">   <Query ID="ALCES\_UKOGBS\_5\_20130212160118" Type="Status" xsi:type="QueryStatus">     <Subscription RepeatTime="5" URL="<http://127.0.0.1:9090/alces/jmf>" >                                <ObservationTarget ObservationPath="\*" />                 </Subscription>                 <StatusQuParams DeviceDetails="Details" JobDetails="Brief" QueueInfo="true" />   </Query> </JMF> |

After Submitting the File to Bambi it will start responding in the give frequency, telling you all the devices states.

[**JDF Editor**](http://community.cip4.org/index.php/tools/jdf-editor)

* 
* 

Description

CIP4 JDF Editor can be used to display, modify, send and receive JDF / [JMF Message](http://jdf4you.org/glossary/5#term21)s to or from [JDF Device](http://jdf4you.org/glossary/5#term27)s. The editor is the tool of choice to check how JDF / JMF works with a specific target device. Using the integrated Send-Option and Http-Server (receive channel) you can send Messages to a target device and check the result with less efforts.

Documentation

To easily run the application, an executable application was created. When the application is initially launched no file is opened and only relevant menu items and buttons are displayed. A file can be opened through a file dialog or through the open recent files menu, where the four most recently opened files are shown. When choosing to open a file, an instance of JDFDocument, a class in the CIP4 JDF Library, is created and the file is parsed through methods in that class. The program checks whether it is a JDF or a JMF file. The file is then sent to a method that creates and displays the tree. The tree is displayed collapsed from start in order to make the opening process less time consuming. The file name is shown in the border of the application. Only one file can be open at a time. If the user chooses to open a completely new file, she/he gets to choose from opening a JDF or JMF file, which gets created through methods in the CIP4 JDF Library.

The automatic validation is switched off by default. The user can switch on and off the automatic validation at any time. If the automatic validation is switched off the validate button is enabled and the user can validate the whole file at any time. When switching it on the whole file is validated. Before opening a file the user can chose to open a file as “read only” by switching off the editing mode.

If an existing file has being modified or a new file has been created and the close or exit button has been pushed the user always gets the question if she/he wants to save the file. The saving of the file is also done through methods in the CIP4 JDF Library.

**The connections between the different views**

In order to be able to navigate the file the error view, the process view and the input and output view are all connected to the main view, the tree view. Clicking on an element in one of these views will result in that the corresponding tree node gets selected. This also works the other way around. Selecting a node in the tree view will load the current data in all the other views, including the attribute view.

Documentation

#### Tree View

The tree view is the main view and it displays the hierarchical structure of the XML file. It also displays the attributes of an element as sub-nodes of the tree node. This will make the attributes easily accessible for modification. To distinct the elements and attributes from each other they are represented with different icons. The tree view has connections to all of the other views in the application in order to make it easy to follow links and still keep track of where in the file the user is currently positioned. Editing is only available from this view and can be done by a right mouse click menu or from the Edit menu.

#### In- & Output View

In the input and output view, either a certain resource with its JDF consumer and producer or a JDF node with the resources it inputs and outputs can be displayed. Clicking on a resource in the tree view will therefore show that particular resource in the center of the view with the JDF node that produces it on the left side and the JDF node that consumes it on the right side. When clicking on a resource link in the tree view, the link is followed to the resource it links to, and that same resource is centered in the view. When clicking on a JDF node, that JDF node will appear in the center, with the input resource(s) on the left and the output resource/resources on the right.

The resources and JDF nodes displayed in this view are displayed as small trees, with their attributes as children, see Figure 3.11. If an element inherits attributes from its parent the text is bold in order to separate them from each other. A left mouse click (which correspond to CTRL and mouse click on Mac) on an element or an attribute in the input and output view has the effect that the corresponding element or attribute gets selected in the tree view. Doing a right mouse click on one of the elements displayed on the sides will make the  element being displayed in the center and the elements it relates to, being displayed on the sides. This feature will help the user to follow links and see how the processes relate to each other.

#### Process View

Another way to follow the processes is in the process view. It displays the process or product a JDF node describes and the sub-processes it contains. Resources are displayed as green rectangles and the JDF nodes as blue rectangles with round corners.

Clicking on a JDF node will always activate the input and output view first. To see the process view the user has to select that tab. Well inside the user can navigate the file. By clicking on the “Go up one level in process view” button located in the button bar, the user can go up to the process that is described by the parent JDF node of the current JDF node. By clicking on a process with the left mouse button the user will expand that process view, hence one level down in the hierarchy. If a JDF node is selected in the tree view, the processes of that node are displayed, otherwise the process view for the top node is displayed. Clicking on a resource or a JDF node in the process view will also select the corresponding node in the tree.

#### Localized Capabilities View                      ???

#### Comment View                                               ???

#### CheckJDF Error View

When the file is validated using the green check mark in the icon bar, the error messages are exposed in the error view, see Figure 3.10. Every error message is connected to the tree node it belongs to. That makes it possible to find where the error occurred and more easily make corrections. The error view is a list that contains elements storing both the error message and the related tree node. The list element is stored in the tree node itself. When a list element in the error view is selected, the appropriate node in the Tree view will also be selected. Double-clicking on an invalid node will let the error view jump to the corresponding list element.

#### Schema Validation View

The validation is done programmatically and is based on algorithms in an existing CIP4 Java application, CheckJDF. There are methods in the CIP4 JDF Library for finding unlinked resources, dangling links, missing attributes etc.

The first thing the application does when a file is being validated is that it checks the whole file for unlinked resources. All errors found from this check are stored in a vector. The tree is built recursively and when a tree node is to be created from the current element, the element is checked for missing elements and attributes, unknown elements and attributes and missing elements and attributes, dangling resource links, missing links among others. The element is also checked against the elements in the vector containing the result from the first check for unlinked resources. If an error is found a Boolean within the tree node is set to be false, which means that it this node is invalid. The tree node gets a special icon in the renderer telling that this element or attribute is invalid. A list element for the error view is created and also stored within the node. If a JDF element has invalid children it shall be displayed as invalid as well, therefore a check is done after the validation of an element to see if the parent element shall be marked as invalid too.

#### DevCap Test Output                                      ???

#### XML Editor

The XML-Editor should better be calles XML-Viewer. It displays the JDF in highlighted plain text.

#### TCPMon

With the TCPMon or TCP Monitor you can keep track of the communication between your MIS and your devices. Set a local Port to which you point your MIS instead of pointing it to the JDF-controller itself. Set the server address and port to the real controller address. The TCPMon will now capture and display all of the communication between them. You can add multiple port/server combinations to monitor different systems. Each connection will be displayed in an own tab.

Each tab has three windows. A list, which lists all of the interactions, and two windows displaying the request and response for each interaction.

#### HTTP server

## [XJDF Libraries](http://community.cip4.org/index.php/tools/xjdf-libraries)

* 
* 

Description

The major conceptual change in JDF 2.0 (or XJDF) is that the technology no longer attempts to model the entire job as one large "job ticket" but rather specifies an interchange format between two applications that are assumed to have an internal data model that is not necessarily based on JDF. Thus each JDF ticket specifies a single transaction between two parties. A single job may be modeled as one or more JDF transactions.

This fact also requires a conceptual change of the JDF libraries as well. Among other things the focus of 1.x libraries is dealing with large XML files (XML files were used as data storage), managing concurrency file access (spawn and merge) and handling the workflow logic (references). Also there are implemented very common XML functionalities like creating Java objects from XSD Schema etc.

The fundamental concepts of JDF (and the library) were designed in the nineties. At this time XML was a very new technology and just a few early XML Tools had existed. Further, the XML technology itself was in a very early phase of development. Therefore, it was essential to include all the implementation specific details mentioned into the JDF Specification to get it working.

Over the last two decades XML has become very popular. Many tools have been designed and many conceptual enhancements in XML have been done. For example, one significant enhancement is XPath. More details about the XPath W3C Standard later in this document or on the W3C website: http://www.w3.org/TR/xpath/.

Due to the major change from JDF 1.x to JDF 2.0 a redesign of the JDF libraries is strongly recommended. The new CIP4 XJDF Library ("xJdfLib") is based on commonly used libraries and technologies like the Apache or JAXB framework. Supplementary functionality which has been removed from JDF Specification no longer makes the library unnecessarily complex for reasons of backwards compatibility. The goal of xJdfLib is to provide a lightweight, modern and easy to use library optimized for actual requirements. This document describes the concepts and the usage of the CIP4 xJdfLib.

## [EasyXJDF](http://community.cip4.org/index.php/tools/easyxjdf)

* 
* 

Description

Tool for XJDF Job Submission.

## [JDF Libraries](http://community.cip4.org/index.php/tools/jdf-libraries)

* 
* 

Description

The CIP4 JDFLibJ Library