# ibrokerKit

**v0.5**

This document describes how to build, configure and install ibrokerKit. It covers both setting up a Community I-Broker and a GRS I-Broker. ibrokerKit contains functionality for registering, operating and managing i-names and i-services.

Community I-Brokers provide i-names with one or more fixed subsegments, e.g. **@free*yourcompany**, or **=web*yourname**.

GRS I-Brokers provide top-level i-names in the central registry, e.g. **@yourcompany** or **=yourname**. GRS I-Brokers need to be accredited by XDI.org. See http://www.inames.net for more information.

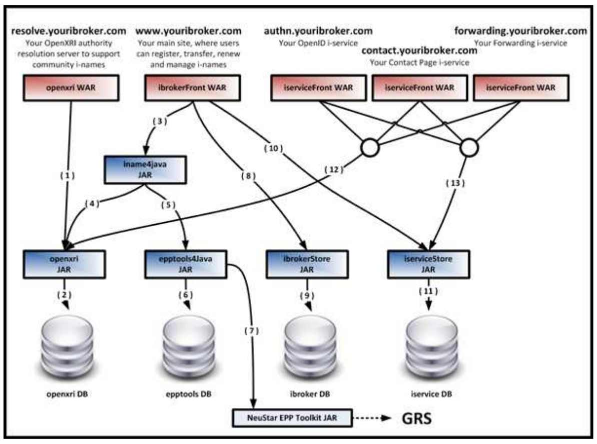
**ChangeLog**

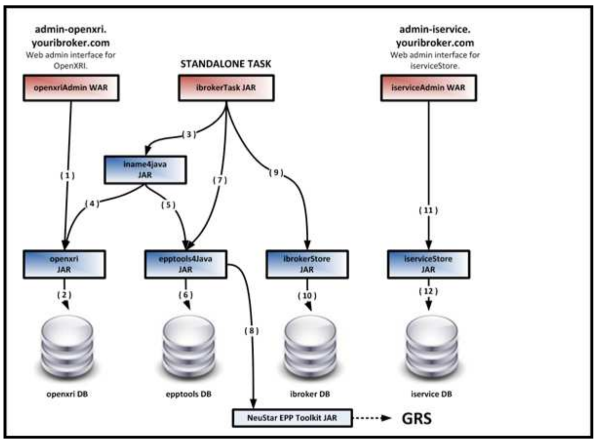
v0.5 2009-01-09, markus@fullxri.com

## Contents

## Introduction

Please take a few minutes to read through http://www.ibrokerkit.com first. This website explains the individual components of ibrokerKit, as well as the relationships between them. The rest of this document assumes some familiarity with the overall ibrokerKit architecture.





## Prerequisites

### All I-Brokers

What you need to use ibrokerKit:

- Java 6.0
- Apache Maven 2.0
- Apache Tomcat 6.0 (or similar servlet container)
- A database server supported by Hibernate (e.g. mySQL)
- A Subversion client

Also recommended:

- Eclipse IDE (http://www.eclipse.org)
- Subclipse Eclipse plugin (http://subclipse.tigris.org)
- Maven Eclipse plugin (http://m2eclipse.codehaus.org)

### Community I-Brokers

Community I-Brokers will need to register and configure the top-level i-names under which they want to offer community i-names. Typically this means adding an Authority Resolution Service Endpoint (SEP) that points to the I-Broker's OpenXRI endpoint, as well as registering the so-called community "root namespaces" in your own OpenXRI server.

For example, if you want to provide community i-names under the **@free** top-level i-name, and if your OpenXRI endpoint is at **resolve.youribroker.com**, then the XRD of **@free** would typically have to contain a SEP like this:

```
<Service priority="10">
  <ProviderID>xri://@free</ProviderID>
  <Type>xri://$res*auth*($v*2.0)</Type>
  <MediaType>application/xrds+xml </MediaType>
  <URI append="qxri" priority="1">https://resolve.youribroker.com/ns/@free/</URI>
</Service>
```

In addition, you need to register the "root namespace" **@free** in your OpenXRI server.

Please refer to the OpenXRI Server User Manual for more information about community i-names.

### GRS I-Brokers

GRS I-Brokers will need the following:

- An organizational i-name and i-number that identifies your I-Broker (e.g. **@youribroker**)
- The EPP Java Toolkit from NeuStar
- A working account at the NeuStar OT&E and/or Production servers

# First Steps

## All I-Brokers

Before installing ibrokerKit you need to do a little planning of your system.

You will need to set up the following databases:

- **ibroker_ibroker**: This is the database for the **ibrokerStore** component. It contains user data.
- **ibroker_openxri**: This is the database for the **OpenXRI** component. It contains XRI subsegments and authorities.
- **ibroker_iservice**: This is the database for the **iserviceStore** component. It contains i-services such as OpenID and Contact Page.

For testing purposes, you could set up these databases on a database server on **localhost**, with the password "ibroker". This is the default setting that you will find in various components in ibrokerKit. In a production environment you should change these settings.

You will need to set up the following hosts:

- **www.youribroker.com**: Your main site. This is where users can register and manage i-names.
- **resolve.youribroker.com**: Your OpenXRI authority resolution server. This is how community i-names get resolved.
- **authn.youribroker.com**: Your endpoint for the OpenID i-service.
- **contact.youribroker.com**: Your endpoint for the Contact Page i-service.
- **forwarding.youribroker.com**: Your endpoint for the Forwarding i-service.
- **locator.youribroker.com**: Your endpoint for the Locator i-service. Optional.
- **admin-openxri.youribroker.com**: Admin interface for the **ibroker_openxri** database. Optional.
- **admin-iservice.youribroker.com**: Admin interface for the **ibroker_iservice** database. Optional.

You can install all of ibrokerKit on a single machine, or you can decide to split up components on different machines. For example, you could decide to have a dedicated database server, a dedicated OpenXRI authority resolution server, etc.

## GRS I-Brokers

GRS I-Brokers will also need to set up the following database in addition to the above:

- **ibroker_epptools**: This is the database for the epptools4java component. It contains EPP transactions with the GRS. This makes the i-broker compliant with section 3.10.2.1 of the I-Broker Agreement of the Global Service Specification at http://gss.xdi.org.

It is highly recommended to use two separate sets of databases (one for OT&E and one for Production GRS operations).

## Getting OpenXRI and ibrokerKit

Now you are ready to check out and build ibrokerKit components. The recommended way to do this is by using the Eclipse IDE with an SVN plugin such as Subclipse.

You should start with a blank workspace. You will check out both OpenXRI and ibrokerKit components into the workspace.

### OpenXRI Repository

*Note by Markus:* This step will get much simpler once we have a new official OpenXRI release.

The following information can be used to check out OpenXRI components:

```
URL: https://openxri.svn.sourceforge.net/svnroot/openxri
Username: -
Password: -
```

You should check out the following projects into your Eclipse workspace:

- openxri4j/trunk
- openxri4j/branches/sandboxes/org.openxri.admin

When checking out openxri4j/trunk, you may want to name the project “openxri” in Eclipse to give it a more meaningful name than “trunk”.

### ibrokerKit Repository

The following information can be used to check out ibrokerKit components:

```
URL: http://ibrokerkit.svnrepository.com/svn/ibrokerkit
Username: ibrokerKit
Password: ibroker8Kit
```

You should check out all projects from this repository.

The ibrokerKit repository contains two types of projects:

- “Component Projects” (e.g. **iserviceStore**, **iname4java**, etc): These are ibrokerKit components. They are either libraries or standalone applications. For each of them there is a corresponding page at http://www.ibrokerkit.com. Normally you want to use these components but not actually modify them.
- “Deployment Projects” (named **com.youribroker.***). These are the actual ibrokerKit web applications that can be configured and deployed.

In addition there is the special **ibrokerKit** project which does not contain any code. It acts as an “umbrella” for other projects and also contains various supporting materials such as docs and distribution builds.

## Building OpenXRI and ibrokerKit

There are some dependencies in ibrokerKit that cannot be resolved automatically by Maven:

- epp.jar: This is the official NeuStar EPP Toolkit. It has to be installed manually before building other components. You should get it from NeuStar during your accreditation process. You can install it into your local Maven repository by executing the following command (make sure Maven 2.0 is in your path):

```
mvn install:install-file -DgroupId=com.neulevel -DartifactId=epp
-Dversion=0.4.9 -Dpackaging=jar -Dfile=epp.jar
```

After that, you can build OpenXRI and ibrokerKit using Maven.

### Building OpenXRI

*Special Note:* You need to make a small change to the "pom.xml" file in the org.openxri.server/ sub-directory of the OpenXRI project (*Note by Markus:* This will be fixed soon): Change the <packaging> setting from "war" to "jar".

Now you need to build OpenXRI and install it into your local Maven repository. Assuming you are in your workspace directory, you can type the following:

```
cd openxri
cd org.openxri.syntax
mvn install
cd ..
cd org.openxri.client
mvn install
cd ..
cd org.openxri.server
mvn install
cd ..
mvn install
cd ..
cd org.openxri.admin
mvn install
```

### Building ibrokerKit

Now you need to build ibrokerKit itself. In your workspace there is one special "umbrella project" named **ibrokerKit**. If you build this project, all other ibrokerKit projects will also be built automatically:

```
cd ibrokerKit
mvn install
```

This step may take a long time, but if you followed the previous steps correctly, it should complete without error.

The “Component Projects” should now have .jar output files in their target/ subdirectories, and the “Deployment Projects” should have .war output files in their target/ subdirectories.

## Configuring ibrokerKit

The “Deployment Projects” in the ibrokerKit SVN repository are complete web applications that run on one of the hosts that make up your I-Broker. E.g. you want to run a web application at **www.youribroker.com** for your main site, and you want to run another web application at **contact.youribroker.com** for your Contact Page i-service. ibrokerKit provides all these web applications, but you have to configure them before (or after) deploying them in your servlet container.

You should copy and rename these Deployment Projects before customizing them. E.g. if you are running an I-Broker at **www.best-inames.com**, then you should copy and rename the **www.youribroker.com** project to **www.best-inames.com**, and then customize it.

Some of these Deployment projects use the Apache Velocity templating engine for customizable appearance. In these cases, the velocity/ subdirectory contains an extra file NOTE.txt that describes the details of these templates. See http://velocity.apache.org for more information about Velocity.

All the Deployment Projects require connections to your databases. The recommended way to set up these connections is to define connection pools in your servlet container, and then reference these connection pools from the individual web applications’ Hibernate settings. Here is an example of what a connection pool can look like in the context.xml file of an Apache Tomcat 6.0 server:

```
<Resource
  name="jdbc/ibroker_openxri"
  auth="Container"
  type="javax.sql.DataSource"
  maxActive="20"
  maxIdle="10"
  maxWait="-1"
  removeAbandoned="true"
  removeAbandonedTimeout="60"
  logAbandoned="true"
  username="ibroker_openxri"
  password="ibroker"
  driverClassName="com.mysql.jdbc.Driver"
  testOnBorrow="true"
  testOnReturn="true"
  testWhileIdle="true"
  validationQuery="SELECT 1"
  url="jdbc:mysql://localhost:3306/ibroker_openxri?autoReconnect=true" />
```

Please refer to the documentation of your servlet container and/or Hibernate for more information on how to set up Hibernate connections. The ibrokerKit projects contains several example files.

## authn.youribroker.com

This is your endpoint for the OpenID i-service.

*velocity/, images/, styles.css:*

These directories and files can be used to configure the appearance (skin).

*WEB-INF/application.properties*

endpoint-path: This is the relative server path of the OpenID endpoint.

endpoint-url: This is the absolute URL of the OpenID endpoint.

(other key/value pairs): All settings can be used as variables in the Velocity templates.

*WEB-INF/iservicestore.properties*

This contains settings for the **iserviceStore** component.

*WEB-INF/openxristore.properties*

This contains settings for the **openxri** component.

## contact.youribroker.com

This is your endpoint for the Contact Page i-service.

*velocity/, images/, styles.css:*

These directories and files can be used to configure the appearance (skin).

*WEB-INF/application.properties*

endpoint-path: This is the relative server path of the Contact Page endpoint.
authn-endpoint-url: This is the absolute URL of the OpenID endpoint.
contact-from: SMTP Sender to use when sending E-Mails.
contact-server: SMTP Host to use when sending E-Mails.
(other key/value pairs): All settings can be used as variables in the Velocity templates.

*WEB-INF/iservicestore.properties*

This contains settings for the **iserviceStore** component.

*WEB-INF/openxristore.properties*

This contains settings for the **openxri** component.

## forwarding.youribroker.com

This is your endpoint for the Forwarding i-service.

*velocity/, images/, styles.css:*

These directories and files can be used to configure the appearance (skin).

*WEB-INF/application.properties*

endpoint-path: This is the relative server path of the Forwarding endpoint.
authn-endpoint-url: This is the absolute URL of the OpenID endpoint.
hxri-prefix: The XRI proxy resolver to use for constructing HXRIs.
(other key/value pairs): All settings can be used as variables in the Velocity templates.

*WEB-INF/iservicestore.properties*

This contains settings for the **iserviceStore** component.

*WEB-INF/openxristore.properties*

This contains settings for the **openxri** component.

## locator.youribroker.com (optional)

This is your endpoint for the Locator i-service.

*velocity/, images/, styles.css:*

These directories and files can be used to configure the appearance (skin).

*WEB-INF/application.properties*

endpoint-path: This is the relative server path of the Forwarding endpoint.
authn-endpoint-url: This is the absolute URL of the OpenID endpoint.
hxri-prefix: The XRI proxy resolver to use for constructing HXRIs.
api-key: An API key for Google Maps, which is used to display a map on the Locator page.
(other key/value pairs): All settings can be used as variables in the Velocity templates.

*WEB-INF/iservicestore.properties*

This contains settings for the **iserviceStore** component.

*WEB-INF/openxristore.properties*

This contains settings for the **openxri** component.

## resolve.youribroker.com

This is your OpenXRI authority resolution server. This is how community i-names get resolved.

*WEB-INF/server.xml*

This is the **openxri** configuration file. Please refer to the OpenXRI Server User Manual for full details about this configuration file.

## admin-openxri.youribroker.com (optional)

This is an admin interface for the **openxri** component.

*WEB-INF/server.xml*

This is the **openxri** configuration file. Please refer to the OpenXRI Server User Manual for full details about this configuration file.

*Warning:* **admin-openxri.youribroker.com** has no built-in security. It is your own responsibility to secure access to this web application by using the Tomcat Realm mechanism, a firewall, or other means.

## admin-iservice.youribroker.com (optional)

This is an admin interface for the **iserviceStore** database.

*WEB-INF/iservicestore.properties*

This contains settings for the **iserviceStore** component.

*Warning:* **admin-iservice.youribroker.com** has no built-in security. It is your own responsibility to secure access to this web application by using the Tomcat Realm mechanism, a firewall, or other means.

## www.youribroker.com (ibrokerFront)

This is your main site. This is where users can register and manage i-names.

For implementing your main site you may choose one of the following options:

- Use **www.youribroker.com (ibrokerFront)** from ibrokerKit as a basis and customize it for your needs.
- Do not use **www.youribroker.com (ibrokerFront)** from ibrokerKit at all and instead implement your main site from scratch, using other ibrokerKit components as dependencies.

The first option is the least amount of work. The second option may make sense if you want maximum flexibility or if you already have a running system into which you want to integrate I-Broker functionality.

**Configuration**

*WEB-INF/application.properties*

This contains various settings for operating **www.youribroker.com**. You will have different settings in this file depending on whether you want to connect to the OT&E or Production GRS servers.

*WEB-INF/server.xml*

This is the **openxri** configuration file. Please refer to the OpenXRI Server User Manual for full details about this configuration file.

*WEB-INF/ibrokerstore.properties*

This contains settings for the **ibrokerStore** component.

*WEB-INF/iservicestore.properties*

This contains settings for the **iserviceStore** component.

*WEB-INF/epptools.properties*

This contains settings for the **epptools4java** component. You will have different settings in this file depending on whether you want to connect to the OT&E or Production GRS servers.

*WEB-INF/ doregister.vm*

A Velocity template for the e-mails sent when a user registers a new i-name. In this template you can use all variables from *WEB-INF/application.properties*, as well as the following:

$iname: The i-name that has been registered

*WEB-INF/ forgotpass.vm*

A Velocity template for the e-mails sent when a user resets their password. In this template you can use all variables from *WEB-INF/application.properties*, as well as the following:

$user: The user’s identifier (normally the first i-name registered by the user)
$email: The user’s e-mail address
$recovery: The recovery code with which the user can reset their password

*WEB-INF/ dorenew.vm*

A Velocity template for the e-mails sent when a user renews one of their i-names. In this template you can use all variables from *WEB-INF/application.properties*, as well as the following:

$iname: The i-name that was renewed

$expdate: The i-name's new expiration date

*WEB-INF/ dotransferin.vm*

A Velocity template for the e-mails sent when a user requests a Transfer IN for one of their i-names. In this template you can use all variables from *WEB-INF/application.properties*, as well as the following:

$iname: The i-name for which the request was issued

$transfertoken: The transfer token which the user needs to approve the transfer at their old i-broker

**EPP Files**

**ibrokerFront** uses the NeuStar EPP Toolkit and therefore requires certain files in its CWD (current working directory):

*Schema files (.xsd)*

*A Keystore file (.jks or .p12)*

*A Truststore file (.jks)*

You need to generate the Keystore file yourself. *Note by Markus: TODO add instructions*

**Payment Processing**

Currently, ibrokerKit has built-in support for the external payment service **2checkout.com**. It can be configured by setting various parameters in the *WEB-INF/application.properties* file. If you need help integrating a different payment solution, please contact us.

## Testing ibrokerKit

You can use the Apache Maven Jetty plugin to conveniently run ibrokerKit web applications locally. For example, if you want to run the **contact.youribroker.com** web application, you can type the following, starting in your workspace:

```
cd com.youribroker.contact
mvn jetty:run
```

If the web application started correctly, you should be able to access it at **http://localhost:8080/com.youribroker.contact/**.

## Standalone Applications

### ibrokerCert

ibrokerCert is a standalone application that is only needed once during the initial Technical Certification phase with NeuStar. It is not used anymore once an I-Broker begins normal operation.

**Configuration**

*conf/application.properties*

host: The IP address of the NeuStar Certification Server
port: The port of the NeuStar Certification Server
clientid: The EPP Client ID to use
password: The EPP Client password to use

When applying for the Technical Certification, you should receive the appropriate values from NeuStar and fill them in.

**EPP Files**

ibrokerCert uses the NeuStar EPP Toolskit and therefore requires certain files in its CWD (current working directory):

*Schema files (.xsd)*

*A Keystore file (.jks or .p12)*

*A Truststore file (.jks)*

All these files are included with ibrokerCert, and you do not have to make any changes to them. Just make sure they are in the CWD when you run the application.

**Debugging Flags**

In the source code file EppCert.java you can set a few flags that control this application. Normally you should not have to actually change them.

*doDumpGreeting*

If true, ibrokerCert will dump the initial EPP Greeting message to a local text file.

*doDumpCommand*

If true, ibrokerCert will dump all sent EPP messages to local text files.

*doDumpResponse*

If true, ibrokerCert will dump all received EPP messages to local text files.

*doChecks*

If true, ibrokerCert will attempt to validate the received EPP messages. This is not recommended as it is not required and may disrupt your certification procedure.

## ibrokerTask

ibrokerTask is a standalone Java application that is meant to be run once a day in the background. It is only needed for GRS I-Brokers, not Community I-Brokers. It performs the following maintenance tasks:

- It polls all incoming events from the I-Broker's GRS queue using EPP. Examples of such events are approved, rejected and canceled i-name transfer requests. ibrokerTask appropriately handles these events.
- It checks all registered i-names for their expiration dates and sends e-mails reminding their owners to renew them.

**Configuration**

*conf/application.properties*

email-from: The SMTP From: header of e-mails sent to users
email-subject: The SMTP Subject: header of e-mails sent to users
email-server: The SMTP server used to send e-mails to users
reminder-days: A list of day differences between today and the expiration date of an i-name. If the difference matches any entry in the reminder-days list, an e-mail will be sent. For example, you can send e-mails 10, 3, 2 and 1 day before a user's i-name expires
(other key/value pairs): All settings can be used as variables in the Velocity templates

*WEB-INF/server.xml*

This is the **openxri** configuration file. Please refer to the OpenXRI Server User Manual for full details about this configuration file.

*WEB-INF/ibrokerstore.properties*

This contains settings for the **ibrokerStore** component.

*WEB-INF/epptools.properties*

This contains settings for the **epptools4java** component. You will have different settings in this file depending on whether you want to connect to the OT&E or Production GRS servers.

*conf/email.vm*

A Velocity template for the e-mails sent to users. In this template you can use all variables from application.properties, as well as the following:
$iname: The i-name that is about to expire
$inumber: The associated i-number
$inamedays: Days before the i-name expires
$inumberdays: Days before the i-number expires

*Warning:* If for whatever reason **ibrokerTask** malfunctions, your users will not get notified of expiring i-names. You should take appropriate measures to monitor **ibrokerTask**, e.g. by recording its logging output, or by running it manually from time to time in addition to running it automatically.

**EPP Files**

**ibrokerTask** uses the NeuStar EPP Toolkit and therefore requires certain files in its CWD (current working directory):

*Schema files (.xsd)*

*A Keystore file (.jks or .p12)*

*A Truststore file (.jks)*

You need to generate the Keystore file yourself. *Note by Markus: TODO add instructions*

## Additional Notes

This is a list of things ibrokerKit does NOT do for you. You will have to take care of these items yourself.

- Make backups of your databases
- GRS I-Brokers: Write policies and fulfill other requirements set forth in the I-Broker Agreement of the Global Service Specification at http://gss.xdi.org.

This list may be incomplete.