D2.5: 1st Integrated Version of FastFix Platform

Lead contractor: TUM
Dennis Pagano, Tobias Roehm, Emítzá Guzmán,

Author(s): Sergio Zamarripa, João García, Benoit Gaudin, Javier Cano, Walid Maalej

Submission date: June 2012
Dissemination level: Public
Abstract: This document describes the first integrated version of the FastFix platform. It is a supplement to the source code, which can be accessed on the FastFix repository[^1]. It gives a conceptual overview of the platform as a whole, and describes how the FastFix platform is typically deployed and used.

[^1]: https://repository.fastfixproject.eu/svn/fastfix/Software/trunk/
Contents

1. Introduction 6

2. Conceptual Overview 7
   2.1. eu.fastfix.client ......................................................... 8
   2.2. eu.fastfix.common ....................................................... 8
   2.3. eu.fastfix.targetapplication ......................................... 8
   2.4. eu.fastfix.server ........................................................ 9
   2.5. eu.fastfix.dependencies ................................................ 9
   2.6. Summary ......................................................................... 9

3. System Manual 10
   3.1. FastFix Client Setup ...................................................... 11
   3.2. FastFix Server Setup ...................................................... 12
   3.3. FastFix Sensors Setup .................................................... 12

4. Summary 13

A. FastFix Development Resources 14

Bibliography 26
List of Figures

2.1. Conceptual Overview of the FastFix Platform ............................. 7
2.2. FastFix Client Platform UI .................................................. 8
3.1. Deployment of FastFix Platform ............................................. 10
3.2. FastFix Client UI ............................................................. 11
3.3. Starting registered sensors via the FastFix Client UI .................... 12
List of Tables

1. Introduction

This document describes the first integrated version of the FastFix platform. It is a supplement to the FastFix source code, which can be accessed in the FastFix source code repository. The document gives a conceptual overview of the platform as a whole, and describes how the FastFix platform is typically deployed and used. For more details on specific aspects and functions of the FastFix platform, we refer the reader to the corresponding deliverables, as summarized in the following list:

- First Prototype of Context Observer
- First Prototype of the User Profiler
- First Prototype of the Error Reporting System
- Refined and Integrated Version of Context Observer, User Profiler and Error Reporting
- First Prototype of the Event Processor
- First Prototype of the Pattern Mining Module
- First Prototype of the Execution Recorder and Replayer
- First and Second Prototypes of the Self-Healing and Patch Generation Component

This document does not describe in detail how to set up a development environment in order to build the FastFix platform from source code. FastFix is a relatively big system, and the project utilizes advanced technologies for the build process. Therefore, in this document we concentrate on giving additional, conceptual information on the source code, and illustrate deployment and usage scenarios. For information on how to start developing for FastFix, we refer the reader to and to the project’s Wiki documentation (attached as Appendix A).

In Section 2, we give a conceptual overview of the integrated FastFix platform. We illustrate how and why bundles are distributed among different namespaces and give details using code metrics. In Section 3, we describe how the integrated platform can be installed and used – in different scenarios (or work modes). Section 4 summarizes and concludes this document.

1 https://repository.fastfixproject.eu/svn/fastfix/Software/trunk/
2 Additionally, we provide the current state of the source code as a snapshot in an FTP site: ftp://ftp.fastfixproject.eu/Materials%20review/FastFix_1st_Integrated_Version
3 http://fastfixproject.eu/wiki/Howto:_Set_up_development_infrastructure
4 http://fastfixproject.eu/wiki/Howto:_Main_development_use_cases
5 http://fastfixproject.eu/wiki/Howto:_Start_FastFix_from_Eclipse
2. Conceptual Overview

FastFix is a remote software maintenance platform. As Figure 2.1 illustrates, its components are distributed among two environments. First, the environment where the target application is running, and second the maintenance environment where development, maintenance, and testing tools are installed. Consequently, the FastFix system consists of two main components: the FastFix client and the FastFix server. Both server and client communicate and exchange data with each other.

![Diagram of FastFix Platform](image)

Figure 2.1.: Conceptual Overview of the FastFix Platform.

As described in [1], we structure the FastFix components into namespaces according to their responsibility in the platform. The namespace of each code bundle can be identified from the bundle name. Currently, the integrated platform contains 65 bundles belonging to 5 different namespaces. In the following, we briefly explain all five namespaces and give additional detail in terms of code metrics. Details on the source code in terms of interfaces and packages can be found in [2].

---

1 Including bundles needed to group other bundles (“parent bundles”)
2 A snapshot of the source code corresponding to this document can be accessed under ftp://ftp.fastfixproject.eu/Materials%202nd%20Review/FastFix_1st_Integrated_Version
3 The project source code repository is located at https://repository.fastfixproject.eu/svn/fastfix/Software/trunk/
2.1. eu.fastfix.client

The client namespace contains bundles constituting the FastFix client component. The FastFix client runs in the environment of the target application (i.e. in the application usage environment). Its main purposes are (a) to collect context information monitored by sensors, (b) to perform data pre-processing tasks before sending information to the maintenance site (for security and performance), (c) to access supervisor mechanisms in the target application (for self-healing), and (d) to provide a user interface to allow users to regulate the FastFix functionality (e.g. start and stop sensors).

The client namespaces currently contains 79 classes with 325 methods in 32 packages (9 bundles), with a total of 2,854 lines of code in 89 files. Figure 2.2 shows a screenshot of the FastFix client UI including three registered sensors.

![FastFix Client Platform UI](image)

Figure 2.2.: FastFix Client Platform UI.

2.2. eu.fastfix.common

The common namespace contains bundles which are needed by both the FastFix client and server. In the current state of the system, the FastFix common namespace includes bundles from nine different areas of concern, including context observation, persistency, error reporting, fault replication, and communication between client and server. More details can be found in [8].

The common namespaces currently contains 134 classes with 964 methods in 39 packages (9 bundles), with a total of 11,040 lines of code in 141 files.

2.3. eu.fastfix.targetapplication

The targetapplication namespace contains bundles that are supposed to run in the runtime environment of the target application. Typically such components are sensors (or actuators). Bundles in this namespace communicate with the FastFix client via the interfaces of the application bridge (cf. [8]).

The targetapplication namespaces currently contains 79 classes with 480 methods in 29 packages (11 bundles), with a total of 4,135 lines of code in 79 files.
2.4. eu.fastfix.server

The *server* namespace contains bundles constituting the FastFix server component. The FastFix server runs in the maintenance environment (or in the application engineering environment). Its main purposes are (a) to collect information sent by FastFix client, (b) to investigate this information and detect performance degradation trends, errors, and possible causes, (c) to create and update user profiles, (d) to provide access to issue trackers, (e) to allow maintenance engineers to replay errors, (f) to create patches and send these patches to clients to self-heal them, and (g) to provide a user interface to allow maintenance engineers to access the FastFix functionality (e.g. error replay and patch generation).

The *server* namespaces currently contains 326 classes with 2,223 methods in 82 packages (16 bundles), with a total of 50,453 lines of code in 328 files.

2.5. eu.fastfix.dependencies

The dependencies namespace contains third-party libraries which we have wrapped in OSGi bundles. This process is sometimes necessary to be able to use specific libraries in an OSGi context. We created OSGi wrappers for the following 8 third party components:

- eu.fastfix.dependencies.axis – wrapping org.apache.axis
- eu.fastfix.dependencies.drools – wrapping org.drools
- eu.fastfix.dependencies.flexjson – wrapping net.sf.flexjson
- eu.fastfix.dependencies.javassist – wrapping javassist
- eu.fastfix.dependencies.jena2 – wrapping com.hp.hpl.jena
- eu.fastfix.dependencies.mysql – wrapping mysql jdbc connector
- eu.fastfix.dependencies.soot – wrapping soot, polyglot, and jasmin

2.6. Summary

Table 2.1 summarizes the code metrics of the first integrated version of the FastFix platform.

<table>
<thead>
<tr>
<th>Module</th>
<th># bundles</th>
<th># packages</th>
<th># classes</th>
<th># methods</th>
<th># LOC</th>
<th># files</th>
</tr>
</thead>
<tbody>
<tr>
<td>eu.fastfix.client</td>
<td>9</td>
<td>32</td>
<td>79</td>
<td>325</td>
<td>2,854</td>
<td>89</td>
</tr>
<tr>
<td>eu.fastfix.common</td>
<td>9</td>
<td>39</td>
<td>79</td>
<td>325</td>
<td>11,040</td>
<td>141</td>
</tr>
<tr>
<td>eu.fastfix.targetapplication</td>
<td>11</td>
<td>29</td>
<td>79</td>
<td>480</td>
<td>4,135</td>
<td>79</td>
</tr>
<tr>
<td>eu.fastfix.server</td>
<td>16</td>
<td>82</td>
<td>326</td>
<td>2,223</td>
<td>50,453</td>
<td>328</td>
</tr>
<tr>
<td>eu.fastfix.dependencies</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sum</td>
<td>53</td>
<td>182</td>
<td>618</td>
<td>3,992</td>
<td>68,482</td>
<td>637</td>
</tr>
</tbody>
</table>

Table 2.1.: Code Metrics of Integrated FastFix Platform.

This section illustrates how the FastFix platform is typically deployed, and which steps are necessary to set up the different components of the FastFix platform.

Figure 3.1.: Deployment of FastFix Platform.

Figure 3.1 illustrates how FastFix is deployed in a typical remote maintenance scenario (components, machines, and connections needed for FastFix are shown in blue). The “Client Machine” refers to any machine hosting parts of the target application. In the case of a simple desktop application, this might just be a standard desktop client. In the case of a three tier application, it might refer to the presentation tier (desktop clients), logic tier (server hosting business logic), and data tier (server hosting database) machines respectively. For FastFix, a “Client Machine” refers to any machine where data about the application, runtime environment, or user can and shall be monitored. To this end, the FastFix client and sensors are deployed onto these “Client Machines”. The “Server Machine” is a (logical) additional machine added to the application deployment scenario, which hosts the FastFix server application. This machine (and the FastFix server) is connected to the clients via internet. The “Tracker Machine” denotes the machine where the issue tracker is running (typically a stand-alone server). The FastFix server connects at runtime to this server in order to access the issue tracker.

The following sections describe on a high level how the FastFix client, server, and sensors have to be installed. Depending on the specific platform and target application the details of these steps may vary, and additional steps may be required. For detailed information on how to set up and test FastFix with the FastFix project scenarios (cf. 9), we refer the reader to the FastFix release notes.

1We put the term “Client Machine” in quotation marks to indicate that these machines are not necessarily clients in terms of the target application language, but in terms of FastFix.

2Note that the FastFix Server application might in theory also be installed on the “Tracker machine” or the “Client machine”.

3https://repository.fastfixproject.eu/svn/fastfix/Software/trunk/deployment/ReleaseNotes/
3.1. FastFix Client Setup

The FastFix client is implemented as an Eclipse RCP (Java) application. The latest binary build can be accessed under http://macbruegge128.informatik.tu-muenchen.de/fastfix/Client/. To run the FastFix client, the following steps have to be accomplished:

1. Go to http://macbruegge128.informatik.tu-muenchen.de/fastfix/Client/

2. Download the latest build of the application and unzip into a folder on the “Client Machine”.

3. Start the client (usually by double-clicking).

4. Specify settings in the preferences menu (currently address of FastFix server and credentials to access the local dataStore).

5. Enable the context observation by clicking on the icon marked with a red circle in Figure 3.2

![FastFix Client UI](image)

**Figure 3.2.: FastFix Client UI.**

After the last step, the FastFix client is ready and waits for sensors to register. As soon as sensors register at the FastFix client, they show up in the UI. From there the sensors can be started, as illustrated in Figure 3.3

---

4 A snapshot has been provided at ftp://ftp.fastfixproject.eu/Materials%20Review/FastFix_1st_Integrated_Version/binaries/Client
5 For details on the sensor lifecycle, we refer the reader to [8]
6 For more details on current FastFix sensors, we refer the reader to [10]
3.2. FastFix Server Setup

The FastFix server is currently implemented as an Eclipse RCP (Java) application, similar as the FastFix client. For future releases, also a console version is planned. The latest binary build can be accessed under http://macbruegge128.informatik.tu-muenchen.de/fastfix/Server/. To run the FastFix server, the following steps have to be accomplished:

1. Go to http://macbruegge128.informatik.tu-muenchen.de/fastfix/Server/.
2. Download and unzip the application into a folder on the “Server Machine”.
3. Start the server (usually by double-clicking).
4. Specify settings in the preferences menu (currently address and credentials of issue tracker and credentials to access the server dataStore).

After the last step, the FastFix server is ready and waits to receive information from clients.

3.3. FastFix Sensors Setup

One main goal of FastFix is to become an open source project, providing a generic maintenance platform. The extensibility of the context observation system is therefore a major concern. The FastFix platform is designed to accept registration and data from arbitrary sensors in the future. The setup procedures for these sensors might be different for different sensors. In general, sensors are first installed, and then register at the FastFix client. From there they can be controlled via the user interface as described in 3.1.

During the FastFix project, several sensors were developed. For a more specific description of these sensors and their usage, we refer the reader to.

---

Footnotes:

7 A snapshot has been provided at ftp://ftp.fastfixproject.eu/Materials%20%202nd%20Review/FastFix_1st_Integrated_Version/binaries/Server

8 A snapshot of the current FastFix sensors has been provided at ftp://ftp.fastfixproject.eu/Materials%20%202nd%20Review/FastFix_1st_Integrated_Version/binaries/Sensors
4. Summary

In this document we gave a conceptual overview of the FastFix platform as a whole, and described how to deploy and use the FastFix platform in typical settings. This document is a supplement to the FastFix source code, which can be accessed in the FastFix source code repository\footnote{https://repository.fastfixproject.eu/svn/fastfix/Software/trunk/}.

\footnote{Additionally, we provide the current state of the source code as a snapshot in an FTP site: ftp://ftp.fastfixproject.eu/Materials%202nd%20Review/FastFix_1st_Integrated_Version}
A. FastFix Development Resources

The following pages contain a copy of three FastFix Wiki pages, describing how to set up a suitable infrastructure to develop for FastFix\(^1\), illustrating how to accomplish main development use cases within this infrastructure\(^2\), and giving detailed instructions on how to build and start the FastFix components on the developers’ machines\(^3\).

\(^1\)From http://fastfixproject.eu/wiki/Howto:_Set_up_development_infrastructure
\(^2\)From http://fastfixproject.eu/wiki/Howto:_Main_development_use_cases
\(^3\)From http://fastfixproject.eu/wiki/Howto:_Start_FastFix_from_Eclipse
Howto: Set up development infrastructure

From FastFix

Infrastructure for FastFix Development

The following software is required to develop FastFix components:

- **Eclipse 3.6** Helios for RCP and RAP Developers (NOT standard Eclipse)  
- **SVN Plugin** for Eclipse from Polarion (Homepage:  
  Eclipse Market Place > Search for "Polarion" > "Subversive" > Install (select all  
  options))
- **Maven 3.0.3** (Homepage: http://maven.apache.org/)
- **M2Eclipse** Maven Plug-In for Eclipse (Homepage: http://m2eclipse.sonatype.org,  
  Install via Updatesite http://m2eclipse.sonatype.org/sites/m2e/: Help > Install New  
  Software > Add... > Use Updatesite URL > Select all options)

---

PLEASE DO NOT USE THE FOLLOWING

We are working on using this M2E version, but it is not working currently. Please use the update M2Eclipse Maven Plug-In for Eclipse (Homepage: http://eclipse.org/m2e/, Install via Updatesite H

Retrieved from "http://fastfixproject.eu/wiki/Howto:_Set_up_development_infrastructure"

Category: Howto

- This page was last modified 11:11, 27 January 2012.
Howto: Main development use cases

From FastFix

This page contains a step by step tutorial on building and running the FastFix Client/Server bundles.

Please refer to D2.2 for a conceptual description of the two roles Developer and Architect.

Contents

- 1 Getting started
  - 1.1 Tutorials
- 2 Developer
  - 2.1 Check-Out Existing Bundles
  - 2.2 Build and Run
    - 2.2.1 Build
    - 2.2.2 Run existing bundles
  - 2.3 Adding new Classes and Interfaces
  - 2.4 Commit
- 3 Architect
  - 3.1 Add new dependency
  - 3.2 Remove dependency

Getting started

- To get started, please set up your infrastructure as described here: Howto: Set up development infrastructure

Tutorials

- OSGi Tutorial: http://www.vogella.de/articles/OSGi/article.html

Developer

A developer may:

- Check out existing bundles
- Build and run existing bundles
- Perform normal source code changes (Write code in existing methods, add classes and interfaces, but NO ARCHITECTURE CHANGES)
- Commit changes

Check-Out Existing Bundles

In order to check out all currently existing bundles, use Subversive SVN for Eclipse (http://www.polarion.com/products/svn/subversive.php). After you prepared your SVN Client

1. Go to the menu Window -> Show View -> Other... and select SVN -> SVN Repositories. The SVN Repository View will appear.
2. Right-click on the view and select New -> Repository Location....
3. In the URL Text field, type in our repository URL https://repository.fastfixproject.eu/svn/fastfix/Software/trunk/.
4. Please provide your authentication information
5. Press Finish.
6. Disable the "Repository Location Properties" dialog with No.
7. From the SVN Repository view select all bundles without selecting the the trunk itsle.
1. Right-click and select "Check-Out".

All bundles should be downloaded to your local Eclipse Package-Explorer. You can check this by changing back to the Eclipse Java Perspective (Window -> Open Perspective -> Java) and looking at the Package Explorer.

**Build and Run**

**Build**

For each platform, client and server, there is a dedicated Maven Project called "parent". This is needed in order to build all required bundles for the client or the server automatically with Maven. The server parent bundle is called "eu.fastfix.server.parent" and the client parent project "eu.fastfix.client.parent". To build the client bundles

1. Right-click on the client parent project and select "Run As... -> Maven Install".
1. Wait for Maven to build the packages and generate the binaries and Manifest.MF files. As a result, you should get something similar to this:

```
<terminated> /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Apr 20, 2011 4:32:50 PM)

[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ eu.fastfix.client.parent ---

[INFO] --- eu.fastfix.client.communication:assembly:run-on-resource [2.50s]
[INFO] --- eu.fastfix.client.applicationBridge:assembly:run-on-resource [1.18s]
[INFO] --- eu.fastfix.client.dataStore:assembly:run-on-resource [1.66s]
[INFO] --- eu.fastfix.client.ontology:assembly:run-on-resource [1.85s]
[INFO] --- eu.fastfix.client.patching:assembly:run-on-resource [1.82s]
[INFO] --- eu.fastfix.client.reporting:assembly:run-on-resource [0.77s]
[INFO] --- fastfix.client.parent:assembly:run-on-resource [0.08s]

[INFO] BUILD SUCCESS

[INFO] Total time: 7.046s
[INFO] Final Memory: 7M/89M
```

1. Refresh your package explorer by pressing F5 or right-click on it and press "Refresh" in order to prevent "Out of sync" while running the bundles.

(Perform analogous steps for the server bundles).

Note: If you have any errors within the Manifest.MF file, please write a ticket in our ticket system. This means that packages or dependencies are not exported or imported properly.

**Run existing bundles**

To run the current bundles

1. Select one of the projects by right-clicking on it
2. Choose "Run As -&gt; Run Configurations...".
3. Double-Click on "OSGi Framework" item from the list placed left. This will create a new configuration file for running the bundles.
4. In the "Arguments tab", replace the proposed "Program arguments" with
   
   1. "-os ${target.os} -ws ${target.ws} -arch ${target.arch} -nl ${target.nl} -console -clean"

5. Press "Run"
6. In the "Console" view, type "ss fastfix" and press enter in order to display all FastFix bundles. If any of the FastFix bundles has "Resolved" state, start it by typing "start xx" where "xx" is the number of the plugin which is not activated.

Note: You only have to create a new configuration and specify arguments once. Later just hit on "Run as OSGi Framework".

**Adding new Classes and Interfaces**

This is an example on how to add a new interface and class to the existing "ApplicationBridge" bundle.

1. Open the project eu.fastfix.client.applicationBridge.
2. Add new Interface "IProvider" in the "applicationBridge" package. (Note: Interfaces should have "I" as prefix, and the real implementation "Impl" as suffix).
3. Add a simple method, for example "public String getEcho(String s);".
4. Create new class "ProviderImpl" inside of "applicationBridge.internal" which implements the interface "IProvider". Note that a real implementation is created inside of the package "applicationBridge.internal". By convention, these packages are not meant to be exported or provided to other packages. See Howto: Inter-Component Communication.
5. After implementing your class, repeat the step for building and running the bundles.

Note: If you have questions on architecture guidelines, refer to D2.2 or the architecture task force (via email or ticket). See also: Howto: Structure FastFix bundles

**Commit**

In order to commit your changes:

1. Select all bundles you have changed.
2. Right-click on them.
3. Choose "Team -&gt; Commit".

If you have any conflicts, please resolve them and repeat the steps.

**Architect**

In addition to developers, an architect may change dependencies of the FastFix project. This includes cross-bundle as well as dependencies to external binaries. Technical Note: By adding or removing a dependency, you change the meta-information of the project (Manifest.MF).

**Add new dependency**

As an example, we will add a dependency to the Jena library to the "eu.fastfix.client.dataStore" plugin.

1. Right-click on the bundle "eu.fastfix.client.dataStore".
2. Go to the "Maven-&gt;Add Dependency..."
3. In the "Add Dependency" Dialog, type "jena" or provide the full name of the library and press OK.

4. M2Eclipse will automatically build a bundle and add Jena as dependency to the project (Technical Note: This affects the file pom.xml).
5. Open the file pom.xml and switch to the "Dependency" tab.
6. Set the scope to "Runtime". With this setting you tell our build system that the Jena library is meant to be used at runtime and not for example for testing.

7. Now you are able to use Jena classes and interfaces within the bundle

Remove dependency

In order to remove a dependency:

1. Open the pom.xml file of the project you want to change
2. Go to the "Dependency tab"
3. Select a dependency and press Delete

Note: DO NOT DELETE "org.eclipse.osgi" DEPENDENCY BECAUSE ALL BUNDLES WE USE ARE OSGI BUNDLES, OTHERWISE YOU WILL GET AN ERROR

Retrieved from "http://fastfixproject.eu/wiki/Howto:_Main_development_use_cases"
This page was last modified 14:04, 27 January 2012.
Howto: Start FastFix from Eclipse

From FastFix

This page explains how to start FastFix client and / or server from Eclipse so that the run configuration actually resembles the binary product.

Contents

- 1 Set-up
- 2 Client
- 3 Server
- 4 In case of errors

Set-up

As first step, you need to check-out the required bundles from the repository:

- eu.fastfix.common.feature (under /deployment/common)
- eu.fastfix.client.feature (under /deployment/client)
- eu.fastfix.client.ui
- eu.fastfix.server.feature (under /deployment/server)
- eu.fastfix.rcp

Client

1. Run a Maven clean install on the common and client parents
2. Select all projects in workspace and do a refresh (by pressing F5 or right-clicking on the project and selecting Refresh)
3. Expand the eu.fastfix.client.ui bundle.

5. Change to the "Dependencies" tab and make sure that there is no error.
6. Change to the "Overview" tab and hit "Synchronize".
7. On the "Overview" tab click the "Validate..." button from the toolbar. There should be no errors or problems.
8. If you have any run configuration for this product delete it. If everything works fine, a correct run configuration should be created automatically when you run the product.
9. Click on "Launch an Eclipse application", or if you want to debug on "Launch an Eclipse application in Debug mode"

Server

1. Run a Maven clean install on the common and server parents
2. Expand the eu.fastfix.server.rcp bundle.
4. Do the same steps as explained in above part about client.

In case of errors

- Make sure that you have all bundles downloaded
- Make sure that the org.eclipse.rcp bundle does not have an error in the dependencies tab of the product file. If it does, it is probably a version conflict. In this case remove it and add org.eclipse.rcp again.

Retrieved from "http://fastfixproject.eu/wiki/Howto:_Start_FastFix_from_Eclipse"

Category: Howto

- This page was last modified 12:45, 8 June 2012.
Bibliography


