



Grant Agreement
 Call: H2020-ICT-2016-2017
 Topic: ICT-13-2016
 Type of action: RIA

No.: 732638
 H2020-ICT-2016-2017
 ICT-13-2016



D2.2: Federation operations, tools and support

Work package	WP 2
Task	Task 2.1 and task 2.2
Due date	31/12//2017
Submission date	10/11/2018
Deliverable lead	Imec
Version	4
Authors	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU)
Reviewers	Peter Van Daele (imec)

Abstract	This deliverable describes the federation operation and support statistics for the first 21 months and the current state of the operational tools.
Keywords	Operations, tools, support, statistics, users, experiments

Document Revision History

D2.2: Federation operations, tools and support

Version	Date	Description of change	List of contributor(s)
V1	15/08/2018	TOC	Brecht Vermeulen (imec)
V2	1/11/2018	First complete version	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU)
V3	8/11/2018	Almost final version	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU)
V4	9/11/2018	Final version	Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU)

DISCLAIMER

The information, documentation and figures available in this deliverable are written by the **Federation for FIRE Plus (Fed4FIRE+)**; project's consortium under EC grant agreement **732638** and do not necessarily reflect the views of the European Commission.

The European Commission is not liable for any use that may be made of the information contained herein.

COPYRIGHT NOTICE

© 2017-2021 Fed4FIRE+ Consortium

ACKNOWLEDGMENT



Co-funded by the
European Union



Co-funded by the
Swiss Confederation

This deliverable has been written in the context of a Horizon 2020 European research project, which is co-funded by the European Commission and the Swiss State Secretariat for Education, Research and Innovation. The opinions expressed and arguments employed do not engage the supporting parties.



D2.2: Federation operations, tools and support

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		R
Dissemination Level		
PU	Public, fully open, e.g. web	<input checked="" type="checkbox"/>
CL	Classified, information as referred to in Commission Decision 2001/844/EC	<input type="checkbox"/>
CO	Confidential to FED4FIRE+ project and Commission Services	<input type="checkbox"/>

* *R*: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

D2.2: Federation operations, tools and support

EXECUTIVE SUMMARY

This deliverable gives an overview of the operational usage of the federation during the first 21 months of the Fed4FIRE+ project. It gives details on statistics of usage, testbeds reachable for the users, how support is organised and statistics on support tickets. It gives also an overview/tutorial/manual of the jFed user tool features.



TABLE OF CONTENTS

DISCLAIMER	2
COPYRIGHT NOTICE	2
ACKNOWLEDGMENT	2
1 STATISTICS ON USERS AND EXPERIMENTS	9
1.1 USERS	9
1.2 EXPERIMENTS	9
1.3 DETAILED MONTHLY USAGE	10
2 SUPPORT FROM WITHIN JFED TOOL	12
2.1 SUPPORT WIZARD IN JFED	12
2.2 SUPPORT REQUESTS BACKEND	16
2.3 OVERVIEW OF SUPPORT REQUESTS	20
2.3.1 'connectivity' support requests	22
2.3.2 'question_jfed' support requests.....	22
2.3.3 'question_testbed' support requests	23
2.3.4 'feature_request' support requests	23
2.3.5 'bug_jfed' support requests	23
2.3.6 'bug_testbed' support requests	23
2.3.7 'bug_user' support requests	23
3 USABLE TESTBEDS	24
4 OVERVIEW NEWEST JFED FEATURES	25
4.1 LINK TEST	25
4.2 CHOOSE A HEALTHY TESTBED FROM JFED	26
4.3 CHOOSE THE RIGHT HARDWARE FROM JFED	27
4.4 GDPR SUPPORT	28
4.5 FUTURE RESERVATION OF RESOURCES	31
4.6 OTHER NEW FEATURES	32
5 APPENDIX: JFED FEATURES, USAGE AND USER MANUAL	33
5.1 PREREQUISITES	33
5.1.1 Windows	33
5.1.2 Linux	33
5.1.3 Mac	33
5.2 INSTALLATION	33
5.3 BASIC FEATURES: FIRST EXPERIMENT TUTORIAL	33
5.3.1 Logging in	33
5.3.2 Create your first experiment	37
5.3.3 Run the experiment.....	39
5.3.4 Login on a node of the experiment	40
5.3.5 Ending the experiment	40
5.3.6 Note on connectivity	41
5.3.7 Test connectivity	42
5.3.8 Feedback and Bug reports in jFed.....	42
5.4 ADVANCED FEATURES	43
5.4.1 Introduction	43
5.4.2 Proxy settings.....	43
5.4.3 Recover slices.....	44
5.4.4 RSpec editor	44
5.4.5 Openflow support.....	45



D2.2: Federation operations, tools and support

5.4.6	Virtual wall advanced features.....	45
5.4.7	Adding extra ssh-keys to nodes	46
5.4.8	Support for subauthorities	47
5.4.9	Ansible support	47
5.4.10	Experiment Specification (espec)	51
5.5	SHORTCUTS	51
5.5.1	General shortcuts.....	51
5.5.2	Experiment definition.....	51
5.5.3	Topology editor	52
5.5.4	RSpec editor	52
5.5.5	Timeline editor.....	52
5.5.6	Experiment.....	52
5.5.7	Topology viewer	53
5.5.8	Timeline viewer	53
5.6	CAVEATS FOR SPECIFIC TESTBEDS.....	53
5.6.1	C-Lab.....	53
5.7	FREQUENTLY ASKED QUESTIONS (FAQ)	54
5.7.1	OS support.....	54
5.7.2	Install problems	54
5.7.3	SSH login problems.....	54
5.7.4	Linux SSH terminal	55
6	OTHER JFED SOFTWARE.....	57
6.1	OVERVIEW	57
6.1.1	GUI tools	57
6.1.2	CLI tools.....	57
6.1.3	Software libraries	58
6.1.4	Fedmon.....	58
6.2	TESTING CONNECTIVITY.....	58
6.2.1	What is the Connectivity Tester ?	58
6.2.2	Installing the Oracle Java JRE and start jFed.....	59
6.2.3	Starting the connectivity test.....	59
6.3	EXPERIMENTER CLI 1 (LEGACY)	60
6.3.1	Overview	60
6.3.2	Configuration.....	61
6.3.3	Download and Run.....	62
6.3.4	Usage	62
6.3.5	Ansible support	67
6.3.6	Speaksfor credential support.....	69
6.3.7	Command Line Help.....	69
6.3.8	Example speaks-for usage	71
6.3.9	Automatic link sharing	72
6.4	EXPERIMENTER CLI 2	72
6.4.1	Overview	72
6.4.2	Configuration.....	74
6.4.3	Download and Run.....	75
6.4.4	Usage	75



LIST OF FIGURES

FIGURE 1: CUMULATIVE ACCOUNTS OVER TIME.....	9
FIGURE 2: NUMBER OF PROJECTS IN THE FEDERATION.....	10
FIGURE 3: NUMBER OF SLIVERS CREATED PER MONTH (THIS CAN BE A NUMBER OF NODES OR A SINGLE)	10
FIGURE 4: NUMBER OF UNIQUE USERS PER MONTH	11
FIGURE 5 JFED FEEDBACK/BUGREPORT BUTTON.....	12
FIGURE 6 START SCREEN OF THE FEEDBACK WIZARD.....	13
FIGURE 7 SPECIFYING WHAT THE QUESTION IS ABOUT	14
FIGURE 8 SPECIFYING WHICH TESTBED THE QUESTION IS ABOUT.....	14
FIGURE 9 ASKING IF THE SUPPORT REQUEST MAY BE POSTED ON A PUBLIC MAILING LIST	15
FIGURE 10 SUPPORT REQUEST DETAILS FORM	16
FIGURE 11 EXAMPLE JIRA TICKET OF A SUPPORT REQUEST	17
FIGURE 12 DESCRIPTION IN JIRA OF A SUPPORT REQUEST.....	18
FIGURE 13 DETAILS-VIEW OF A SUPPORT REQUEST.....	19
FIGURE 14: DETAILED OVERVIEW OF A SINGLE API CALL	19
FIGURE 15 SUPPORT REQUESTS BY TYPE	21
FIGURE 16 SUPPORT REQUESTS BY TYPE AND OVER TIME	22
FIGURE 17: OVERVIEW OF AVAILABLE TESTBEDS FROM THE FEDERATION MONITOR VIEW.....	24
FIGURE 18: OVERVIEW OF THE AVAILABLE TESTBEDS FROM THE JFED VIEW	24
FIGURE 19: JFED NETWORK EXPERIMENT FOR LINK TESTING (LINK TEST BUTTON CAN BE SEEN ON TOP MIDDLE).....	25
FIGURE 20: JFED LINK TEST RESULTS, INCLUDING A SPEED TEST	26
FIGURE 21: TESTBED AND RESOURCE AVAILABILITY VISIBLE IN JFED	27
FIGURE 22: JFED AVAILABILITY PER HARDWARE TYPE.....	28
FIGURE 23: STARTING AN EXPERIMENT WITH A TESTBED NEEDING GDPR COMPLIANCE CHECK.....	29

D2.2: Federation operations, tools and support

FIGURE 24: A SPECIFIC (EXAMPLE) PAGE PER TESTBED FOR TERMS AND CONDITIONS, OPENS IN THE BUILT-IN BROWSER OF JFED.....	29
FIGURE 25: SAME WINDOW, NOW AFTER APPROVAL. NOTICE ALSO THE REVOKE BUTTON	30
FIGURE 26: AFTER APPROVAL OF THE TERMS AND CONDITIONS	31
FIGURE 27: OPENING A TESTBED ADVANCE RESERVATION WEBSITE FROM JFED31	
FIGURE 28: OPEN A WEBBROWSER TO A NODE, FROM JFED	32



1 STATISTICS ON USERS AND EXPERIMENTS

1.1 USERS

The figure below shows the cumulative number of accounts in the federation since September 2013. The previous project Fed4FIRE and the current Fed4FIRE+ project are indicated. In total there have been created more than 1200 user accounts and on top of that about 350 accounts for classes have been made. These class accounts are typically reused each year.

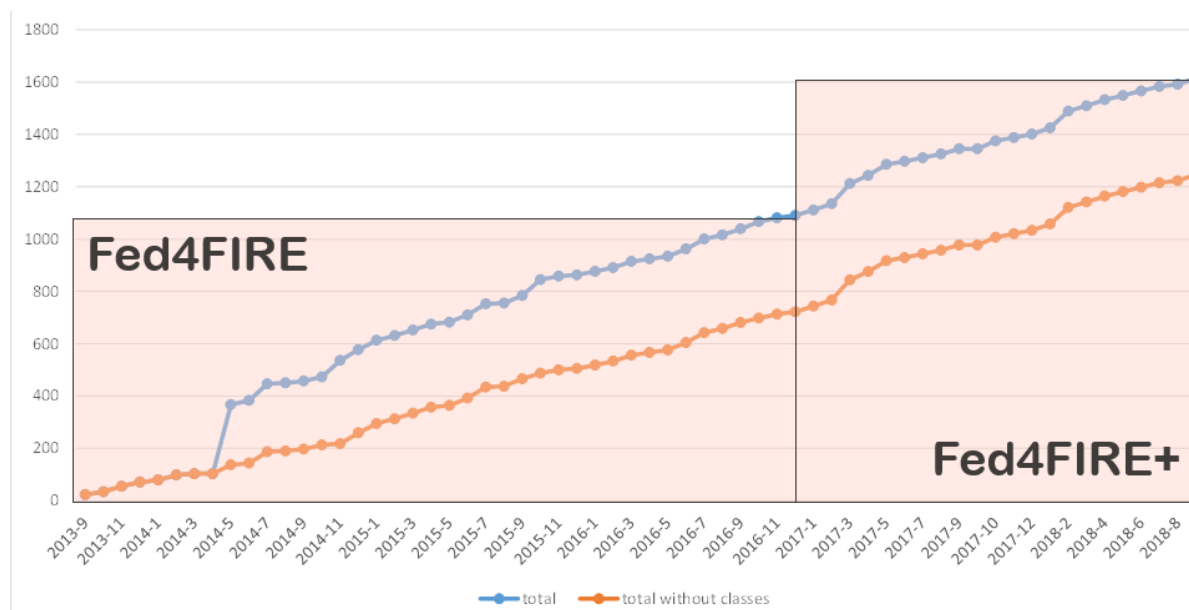


Figure 1: Cumulative accounts over time

1.2 EXPERIMENTS

The graph below gives an answer to the question 'how many experiments have been run?'. For this, we have the concept of a 'project' in the Fed4FIRE authority. A project is requested by a PI (Principal Investigator) and can contain multiple people. Examples of projects are e.g. a PhD, a master student thesis, a research project, an open call experiment. It is clear that within such a project (e.g. a PhD over multiple years) multiple tests/setups are run on the same topic. From September 2013 till now, about 650 projects have been created. Interesting to note is that during Fed4FIRE 46 Open Call experiments were run, and during Fed4FIRE+ till now about 40. It shows that this is a minority compared to all other experiments.

D2.2: Federation operations, tools and support

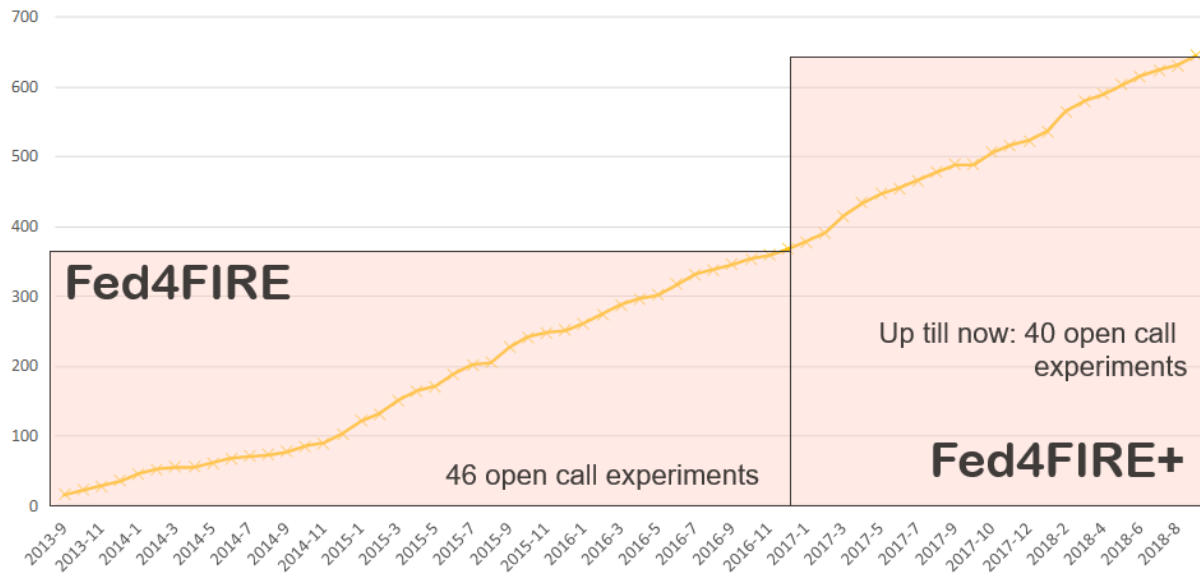


Figure 2: Number of projects in the federation

1.3 DETAILED MONTHLY USAGE

The previous graphs are cumulative, so the question arises what the actual usage is. The following two graphs illustrate this. A slice exists out of slivers. A slice is the collection of resources a user reserves and provisions to run his experimentation. A sliver is a technical concept and varies per testbed: on some testbeds a sliver can exist out of multiple nodes, on other testbeds each node in an experiment is a sliver. A sliver consists always of resources of a single testbed. The graph below shows per month the number of slivers created. In some months peaks of more than 3000 created slivers can be seen. The feature in jFed to store the slivers, was introduced in the stable version end of 2015. That's the reason before that only moderate usage can be seen.

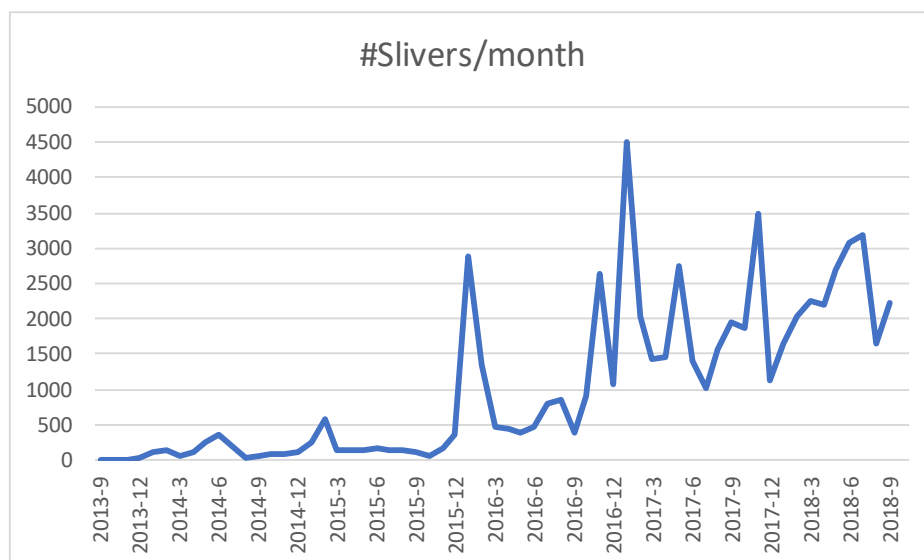


Figure 3: Number of slivers created per month (this can be a number of nodes or a single)

D2.2: Federation operations, tools and support

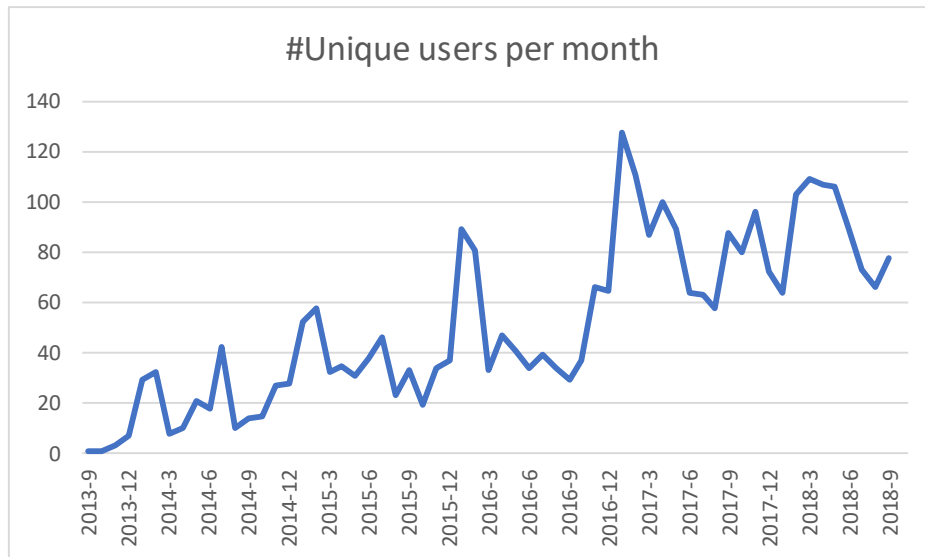


Figure 4: Number of unique users per month

Figure 4 gives an indication of the number of unique users reserving resources each month.

To verify that these users are not the same each month, we also verified the number of unique users during the period from March 2018 till September 2018: 238.

We also verified the average sliver duration: 96 hours.

2 SUPPORT FROM WITHIN JFED TOOL

2.1 SUPPORT WIZARD IN JFED

Starting from jFed 5.7.2, which was released on March 22th, 2017, the jFed Experimenter GUI contains an advanced support wizard. This wizard allows the user to contact the Fed4FIRE+ support team at imec for assistance when they run into problems while using one or more testbed resources.

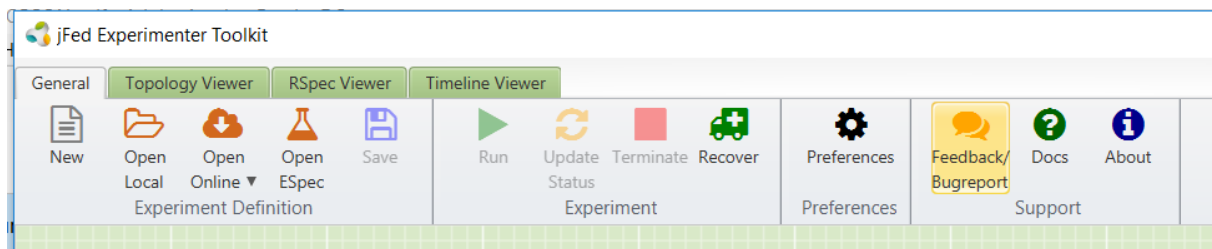
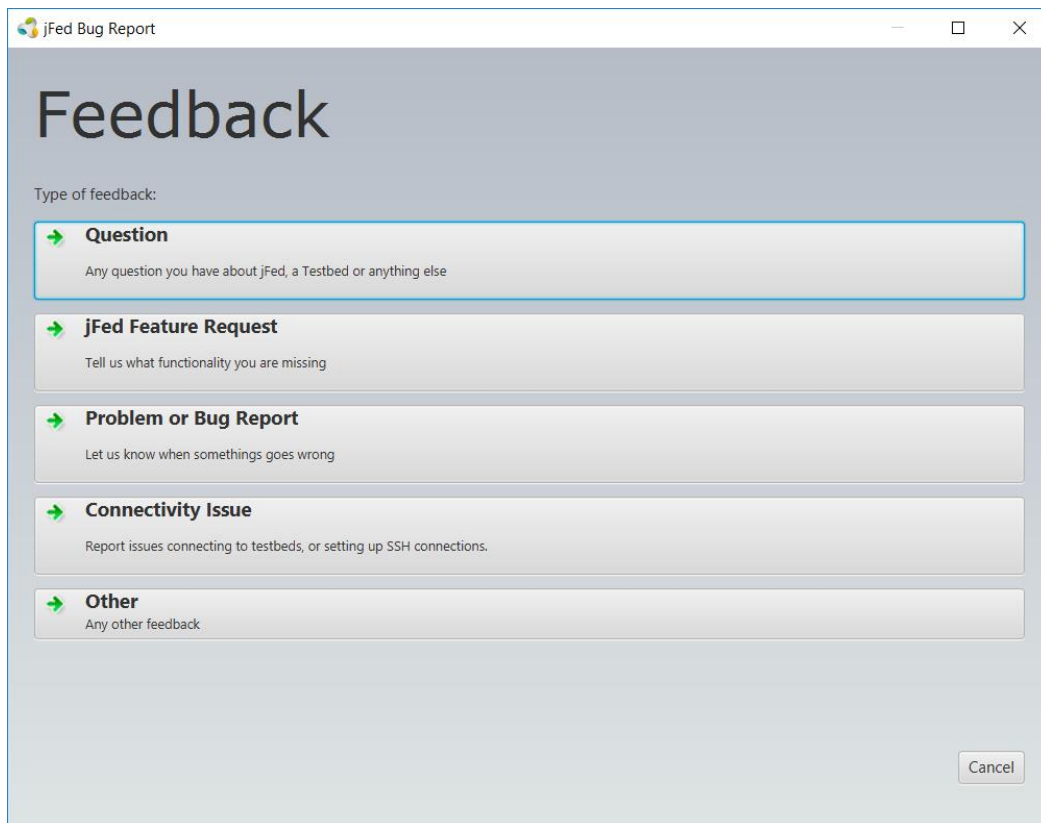


Figure 5 jFed Feedback/Bugreport button

The first page of the wizard allows the user to select the category for the question he wants to pose:

- **Question:** any question about jFed, a testbed or anything else
- **jFed Feature Request:** for reporting missing functionality in the jFed GUI
- **Problem or Bug Report:** for reporting unexpected behaviour of the jFed GUI or a testbed
- **Connectivity issue:** for when the user is unable to connect to a testbed API endpoint or a testbed resource
- **Other:** any other feedback

D2.2: Federation operations, tools and support



The screenshot shows a window titled "jFed Bug Report" with a "Feedback" header. Below the header, it says "Type of feedback:" followed by five selectable options, each with a green arrow icon:

- Question**: Any question you have about jFed, a Testbed or anything else
- jFed Feature Request**: Tell us what functionality you are missing
- Problem or Bug Report**: Let us know when somethings goes wrong
- Connectivity Issue**: Report issues connecting to testbeds, or setting up SSH connections.
- Other**: Any other feedback

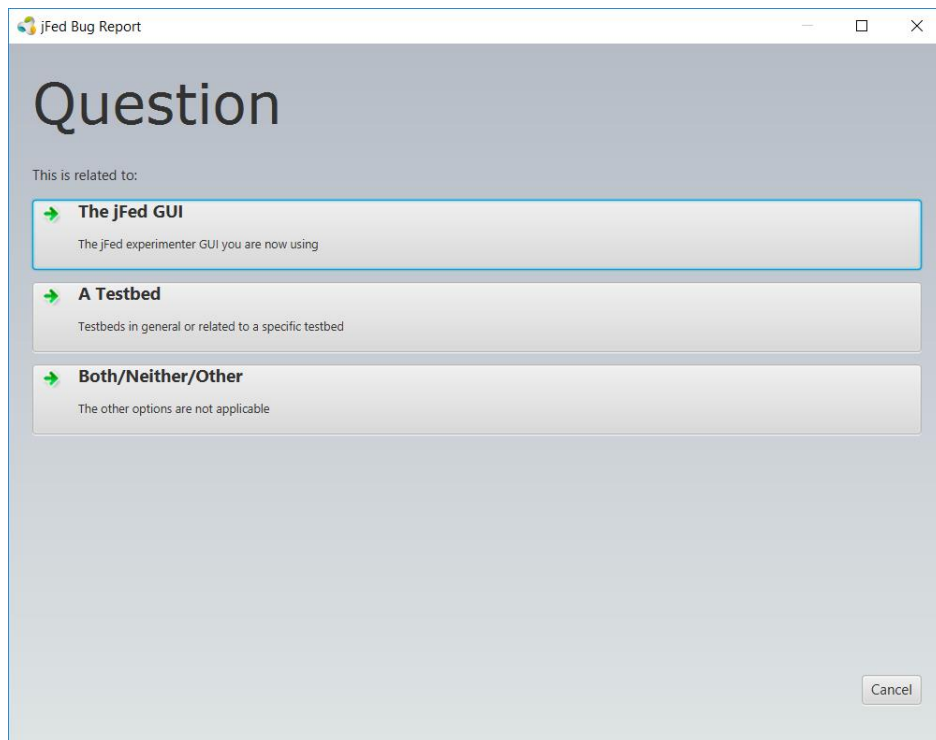
A "Cancel" button is located in the bottom right corner of the window.

Figure 6 Start screen of the Feedback wizard

Depending on the category the user chooses, he has the option to further specify the type of support request he has. The screenshots below show the follow-up pages when an user selects the category 'Question'. The first follow-up question allows him to specify if he has a question about the jFed GUI and/or a testbed that he/she wants to use. When the user selects the latter, a second follow-up question allows the user to select the testbeds about which the question goes. For ease of use, the testbeds that are currently active in an experiment are put on top of the list, and are indicated in bold.

By asking these questions, it is possible to do a first triage when the support request is submitted. This allows the support backend to automatically forward the request to the correct testbed support contact.

D2.2: Federation operations, tools and support



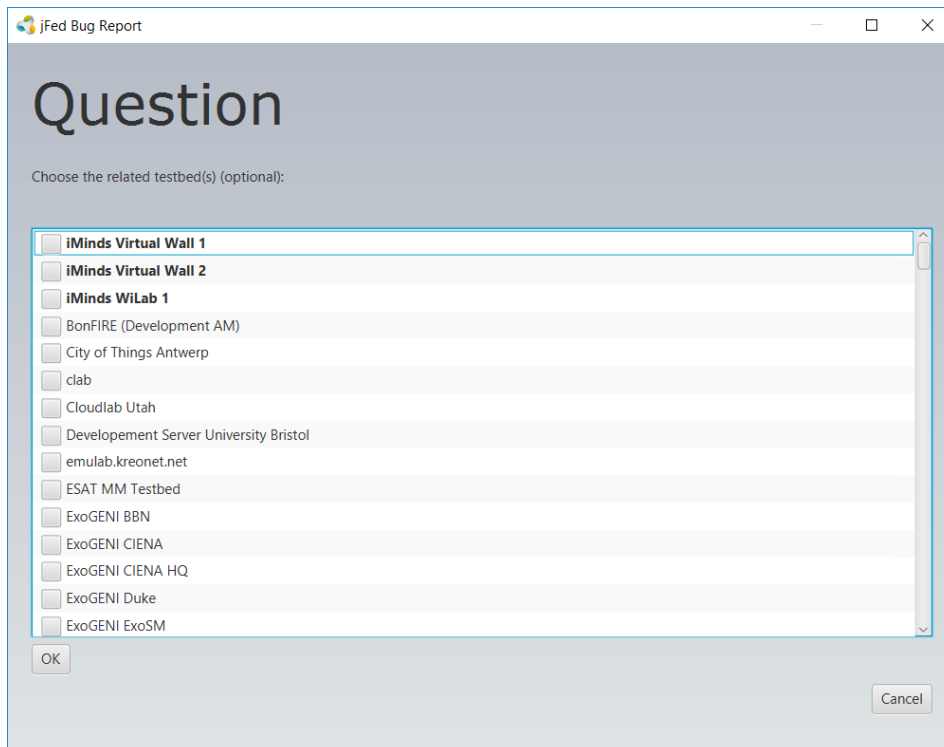
Question

This is related to:

- The jFed GUI**
The jFed experimenter GUI you are now using
- A Testbed**
Testbeds in general or related to a specific testbed
- Both/Neither/Other**
The other options are not applicable

Cancel

Figure 7 Specifying what the question is about



Question

Choose the related testbed(s) (optional):

- iMinds Virtual Wall 1
- iMinds Virtual Wall 2
- iMinds WILab 1
- BonFIRE (Development AM)
- City of Things Antwerp
- clab
- Cloudlab Utah
- Development Server University Bristol
- emulab.kreonet.net
- ESAT MM Testbed
- ExoGENI BBN
- ExoGENI CIENA
- ExoGENI CIENA HQ
- ExoGENI Duke
- ExoGENI ExoSM

OK

Cancel

Figure 8 Specifying which testbed the question is about

When appropriate, the support form asks the user if he wants the question to also be posted on the public Fed4FIRE+ experimenters mailing list. This encourages users to share their

D2.2: Federation operations, tools and support

problems with other experimenters, and creates a publicly accessible list of frequently asked questions and their respective solutions.

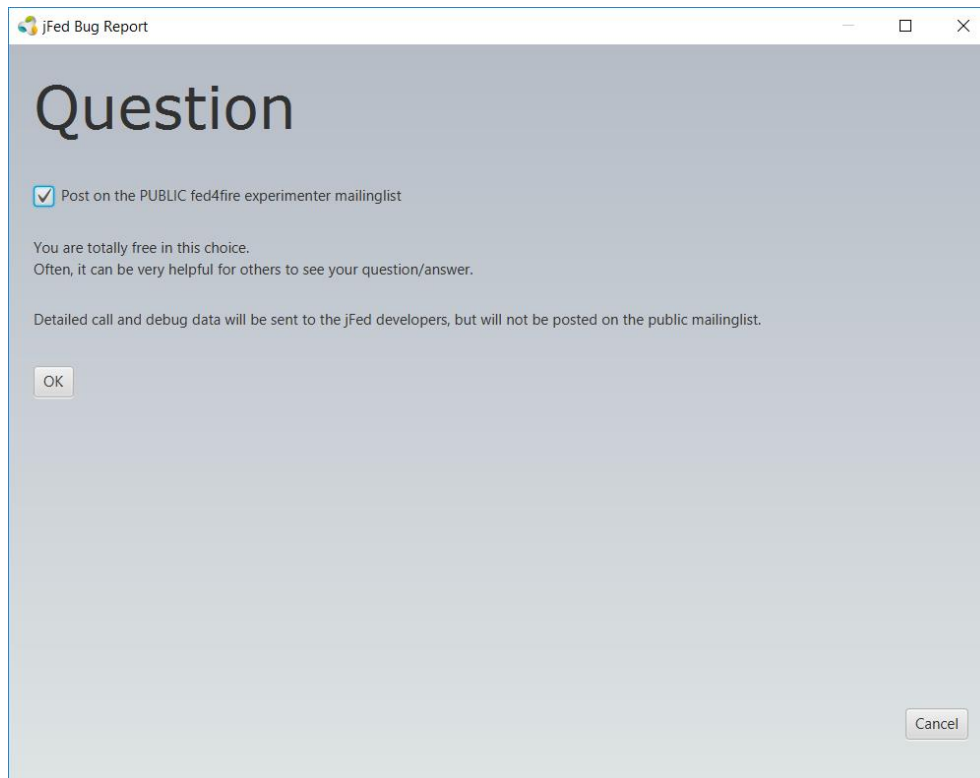


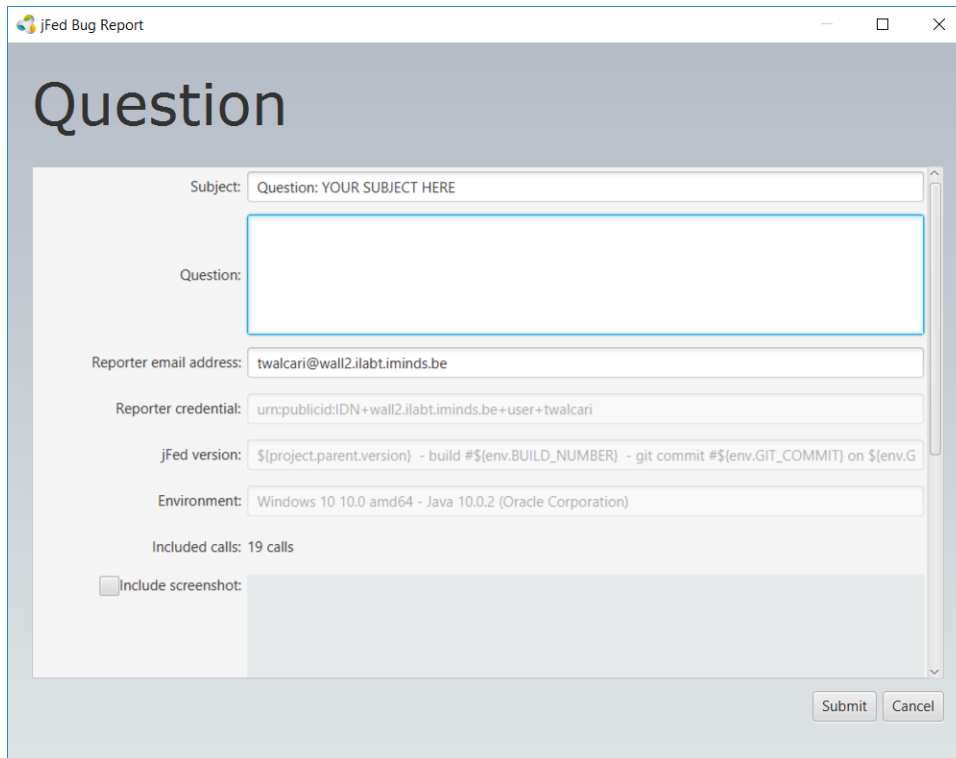
Figure 9 Asking if the support request may be posted on a public mailing list

The last screen of the wizard allows the user to enter his question, and shows which information will be included in the support request. This information always contains:

- An unique ID of the experimenter “reporter credential”
- The email address on which the experimenter can be reached
- The jFed version used to submit the support request
- The environment in which jFed runs (OS and Java version)
- All API calls made by jFed up until that point (request and responses)

Optionally the user can also include a screenshot to help explain his support request.

D2.2: Federation operations, tools and support



The screenshot shows a web browser window titled "jFed Bug Report" with a "Question" form. The form contains the following fields and elements:

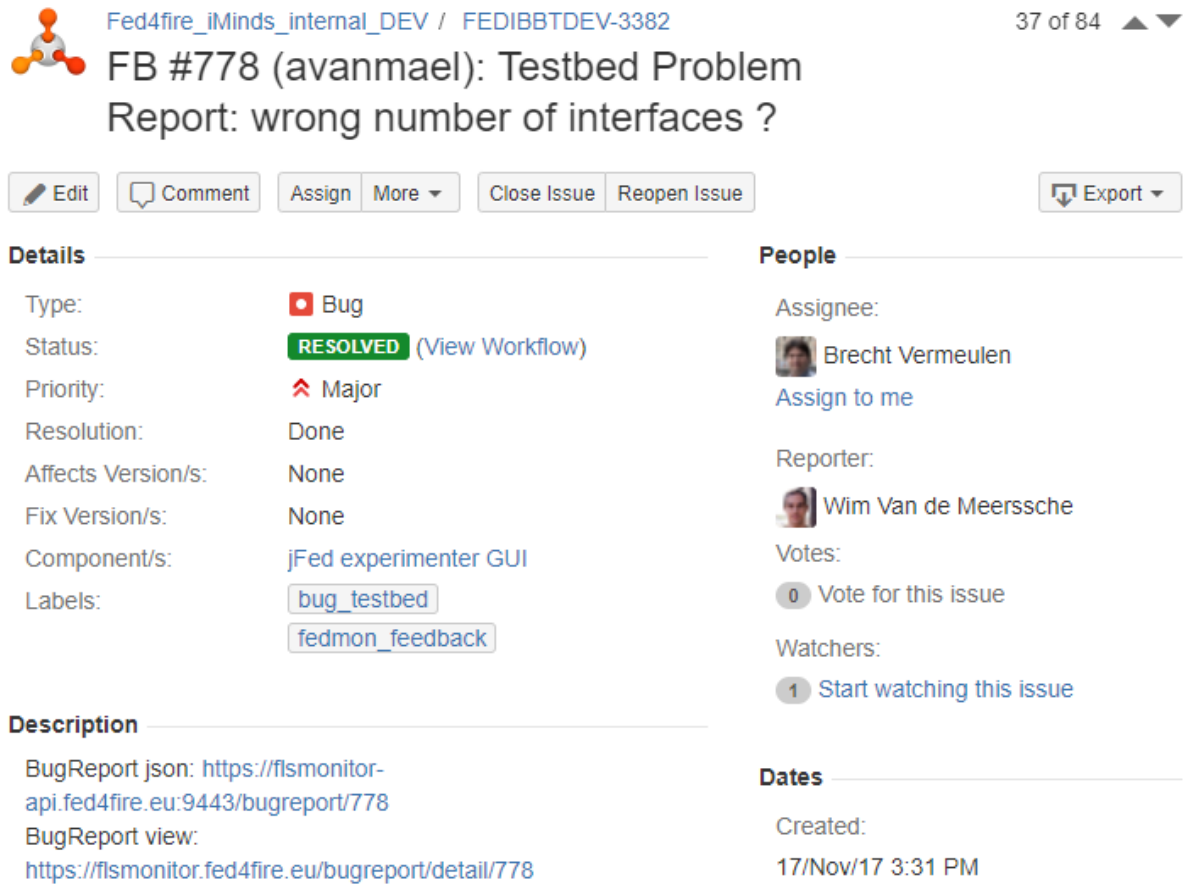
- Subject:** A text input field containing "Question: YOUR SUBJECT HERE".
- Question:** A large, empty text area for the user's question.
- Reporter email address:** A text input field containing "twalcari@wall2.ilabt.iminds.be".
- Reporter credential:** A text input field containing "urn:publicid:DN+wall2.ilabt.iminds.be+user+twalcari".
- jFed version:** A text input field containing a template string: "\${project.parent.version} - build #\${env.BUILD_NUMBER} - git commit #\${env.GIT_COMMIT} on \${env.GIT_COMMIT}".
- Environment:** A text input field containing "Windows 10 10.0 amd64 - Java 10.0.2 (Oracle Corporation)".
- Included calls:** A text input field containing "19 calls".
- Include screenshot:** A checkbox that is currently unchecked.
- Buttons:** "Submit" and "Cancel" buttons are located at the bottom right of the form.

Figure 10 Support request details form

2.2 SUPPORT REQUESTS BACKEND

The submitted support requests automatically create a ticket in a JIRA issue tracker, and are also sent as a mail to the Fed4FIRE+ support team in imec.

D2.2: Federation operations, tools and support



The screenshot shows a JIRA ticket interface. At the top right, there is a '37 of 84' indicator with up and down arrows. The ticket title is 'FB #778 (avanmael): Testbed Problem Report: wrong number of interfaces ?'. Below the title are several action buttons: 'Edit', 'Comment', 'Assign', 'More', 'Close Issue', 'Reopen Issue', and 'Export'. The 'Details' section on the left lists: Type: Bug, Status: RESOLVED (View Workflow), Priority: Major, Resolution: Done, Affects Version/s: None, Fix Version/s: None, Component/s: jFed experimenter GUI, and Labels: bug_testbed, fedmon_feedback. The 'People' section on the right lists: Assignee: Brecht Vermeulen (Assign to me), Reporter: Wim Van de Meerssche, Votes: 0 (Vote for this issue), and Watchers: 1 (Start watching this issue). The 'Description' section on the left contains two links: 'BugReport json: https://flsmonitor-api.fed4fire.eu:9443/bugreport/778' and 'BugReport view: https://flsmonitor.fed4fire.eu/bugreport/detail/778'. The 'Dates' section on the right shows 'Created: 17/Nov/17 3:31 PM'.

Figure 11 Example JIRA ticket of a support request

The description of the support request contains (links to) all the info the user provided.

D2.2: Federation operations, tools and support

Description

BugReport json: <https://flsmonitor-api.fed4fire.eu:9443/bugreport/771>

BugReport view: <https://flsmonitor.fed4fire.eu/bugreport/detail/771>

User email: avanmael@wall2.ilabt.iminds.be✉

User URN: urn:publicid:IDN+wall2.ilabt.iminds.be+user+avanmael (Wall2 Users)

Feedback:

- Type: Bug Report
- About: Testbed(s)
- Related testbeds:
 - urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm
- Description:
 - on node n0710-10.wall1.ilabt.iminds.be, eth5 doesn't seem to be connected to the LAN as it cannot ping any other node
- jFed Version: 5.8.0 - build #4 - git commit #05fa10136faf211b90d49d0eb2e92bc03df36a5a on HEAD
- jFed Environment: Windows 7 6.1 amd64 - Java 1.8.0_144 (Oracle Corporation)
- Send to mailinglist: — No
- Included calls: 65
- Includes screenshot: No

Figure 12 Description in JIRA of a support request

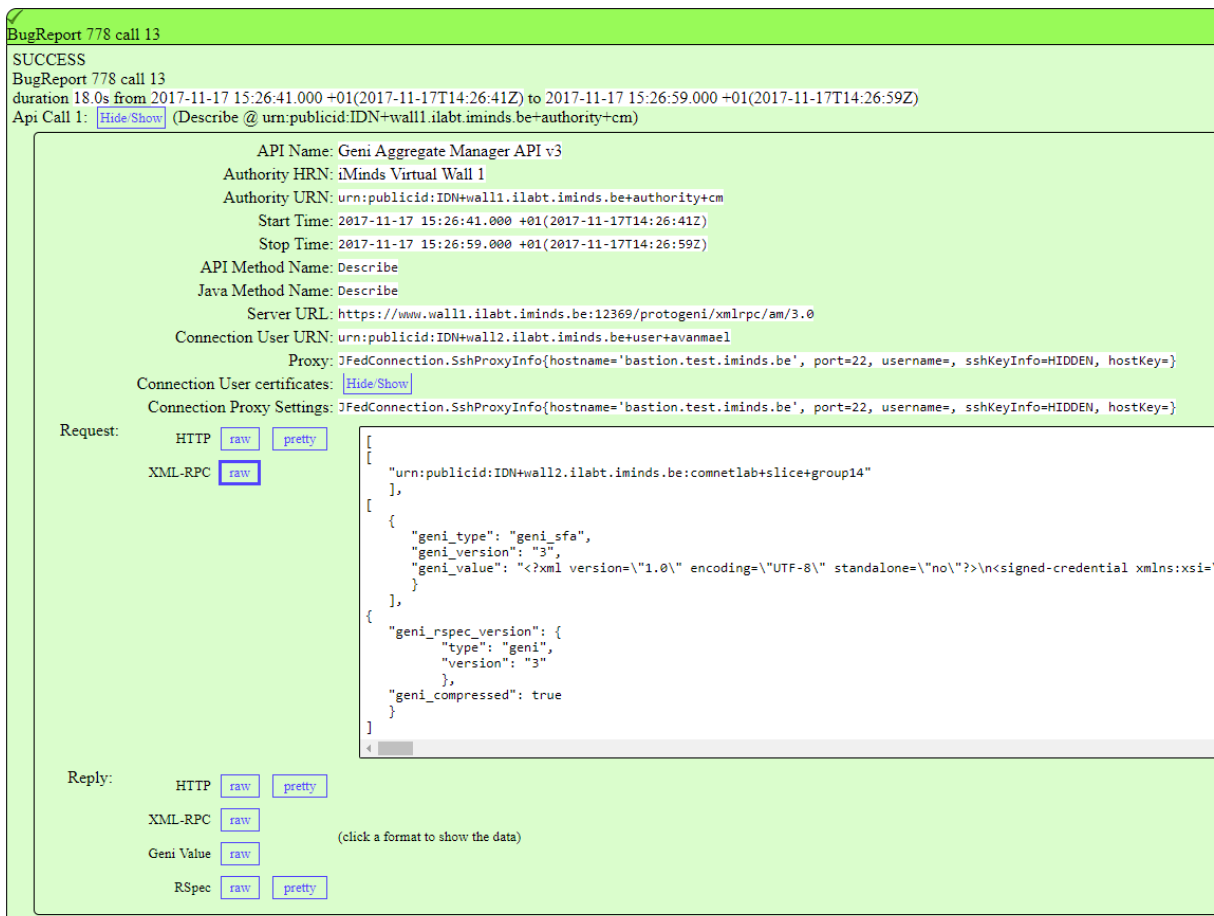
The description also contains links to a more detailed bugreport-view which exposes – amongst others – all API requests and responses done by the jFed GUI, the results of a connectivity test, and the last 2048 log lines generated by the jFed software:

D2.2: Federation operations, tools and support

Details:

ID: 771
 Reporter Urn: urn:publicid:IDN+wall2.ilabt.iminds.be+user+avanmael
 Date: 2017-11-16T11:54:07Z
 reportType: BUG
 reportTarget: TESTBED
 relatedTestbeds: urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm
 subject: Testbed Problem Report: issue on interface
 postOnPublicList: false
 Description: on node n0710-10.wall1.ilabt.iminds.be, eth5 doesn't seem to be connected to the LAN as it cannot ping any other node
 screenshot: None
 jFed Version: 5.8.0 - build #4 - git commit #05fa10136faf211b90d49d0eb2e92bc03df36a5a on HEAD
 User Info: (Click to show)
 Email Address: avanmael@wall2.ilabt.iminds.be
 preferences: (Click to show)
 API Calls: (Click to show 65 Api Call URLs)
 connectivityTestResults: (Click to show 191 connectivity test results)
 logLines: (Click to show 2048 loglines)
 Slices: (Click to show 19 slices)
 stitchingJobReports:

Figure 13 Details-view of a support request



BugReport 778 call 13
 SUCCESS
 BugReport 778 call 13
 duration 18.0s from 2017-11-17 15:26:41.000 +01(2017-11-17T14:26:41Z) to 2017-11-17 15:26:59.000 +01(2017-11-17T14:26:59Z)
 Api Call 1: [Hide/Show](#) (Describe @ urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm)

API Name: Geni Aggregate Manager API v3
 Authority HRN: iMinds Virtual Wall 1
 Authority URN: urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm
 Start Time: 2017-11-17 15:26:41.000 +01(2017-11-17T14:26:41Z)
 Stop Time: 2017-11-17 15:26:59.000 +01(2017-11-17T14:26:59Z)
 API Method Name: Describe
 Java Method Name: Describe
 Server URL: https://www.wall1.ilabt.iminds.be:12369/protogeni/xmlrpc/am/3.0
 Connection User URN: urn:publicid:IDN+wall2.ilabt.iminds.be+user+avanmael
 Proxy: JFedConnection.SshProxyInfo{hostname='bastion.test.iminds.be', port=22, username=, sshKeyInfo=HIDDEN, hostKey=}
 Connection User certificates: [Hide/Show](#)
 Connection Proxy Settings: JFedConnection.SshProxyInfo{hostname='bastion.test.iminds.be', port=22, username=, sshKeyInfo=HIDDEN, hostKey=}

Request:
 HTTP [raw](#) [pretty](#)
 XML-RPC [raw](#)

```

[
  {
    "urn:publicid:IDN+wall2.ilabt.iminds.be:comnetlab+slice+group14"
  },
  [
    {
      "geni_type": "geni_sfa",
      "geni_version": "3",
      "geni_value": "<?xml version='1.0' encoding='UTF-8' standalone='no'?'>\n<signed-credential xmlns:xsi="
    },
    {
      "geni_rspec_version": {
        "type": "geni",
        "version": "3"
      },
      "geni_compressed": true
    }
  ]
]
  
```

Reply:
 HTTP [raw](#) [pretty](#)
 XML-RPC [raw](#)
 Geni Value [raw](#)
 RSpec [raw](#) [pretty](#)

(click a format to show the data)

Figure 14: Detailed overview of a single API call

D2.2: Federation operations, tools and support

All subsequent actions taken for processing the support request are logged into the JIRA-ticket. This allows the Fed4FIRE+ support team at imec to follow-up on issues and ensure a quick and thorough solution for the experimenter.

2.3 OVERVIEW OF SUPPORT REQUESTS

Starting from the release of jFed 5.7.2 on March 22th, 2017; we received 243 support requests via this form. There were about 1200 reported issues by 190 different users (of which 10 US GENI users which reported issues). Of these 1200, about 900 are uncaught exceptions which are automatically reported by jFed after approval by the user and help us to improve jFed. The 243 support requests are initiated by humans. After manually verifying the contents of these support requests (and removing the uncaught exceptions), we can break them down into 7 categories:

- **connectivity**: the user is experiencing a network problem preventing him from reaching one or more Fed4FIRE+ testbed API endpoints and/or resources on a testbed
- **question_jfed**: the user has a problem using jFed which was resolved by explaining a jFed feature and/or directing him to the relevant jFed documentation
- **question_testbed**: the user has a problem using a testbed which was resolved by explaining how a testbed functions and/or by directing him to the relevant testbed documentation
- **feature_request**: a request for a new feature in the jFed GUI
- **bug_jfed**: the user experienced unexpected behaviour by the jFed GUI, which had to be resolved with a bugfix in jFed
- **bug_testbed**: the user experienced unexpected behaviour by a testbed, which had to be resolved by the testbed support operator
- **bug_user**: the user experienced unexpected behaviour because of an error at the end-user end (wrong PC-clock, incorrect RSpec, broken Java setup, ...)

D2.2: Federation operations, tools and support

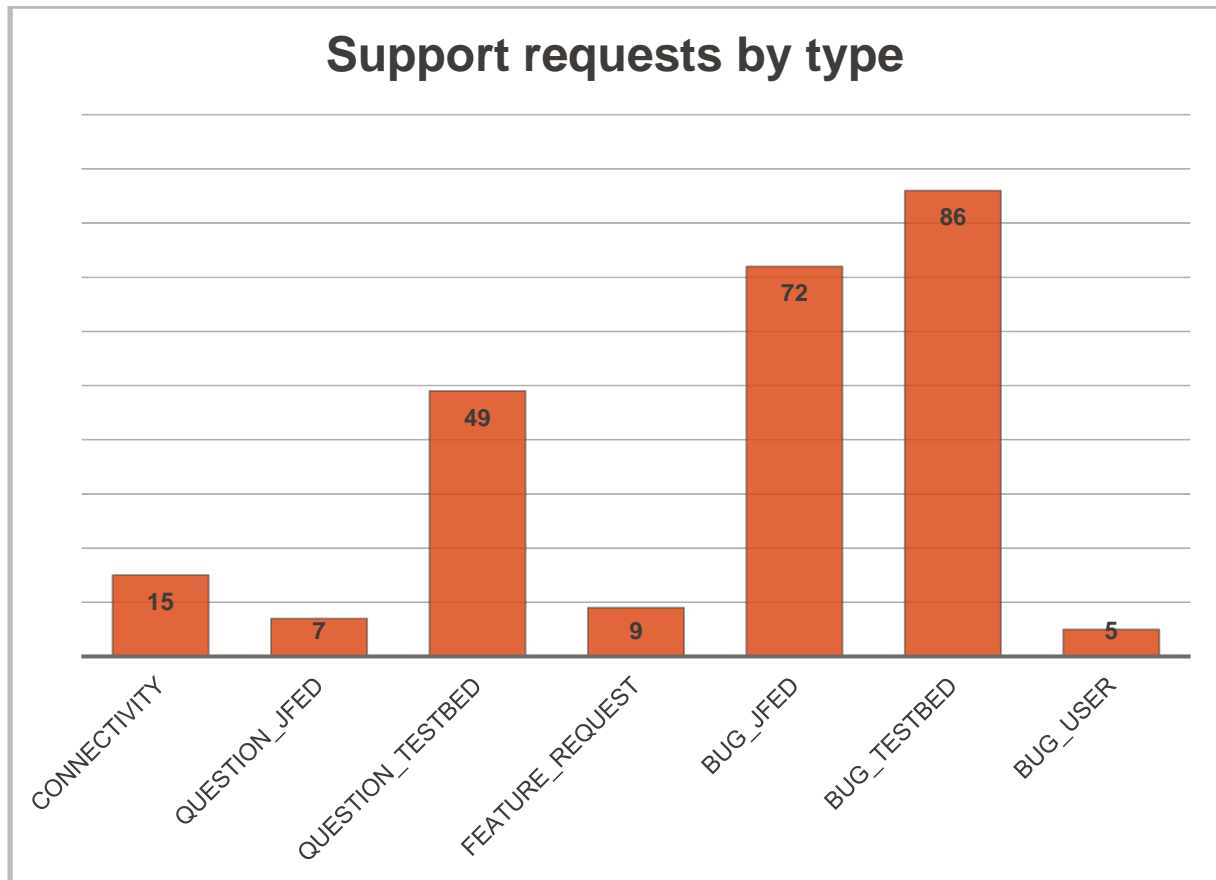


Figure 15 Support requests by type

To end, Figure 16 gives an overview of the support requests over time. It can be seen from this, that usage and support requests are spread over time.

D2.2: Federation operations, tools and support

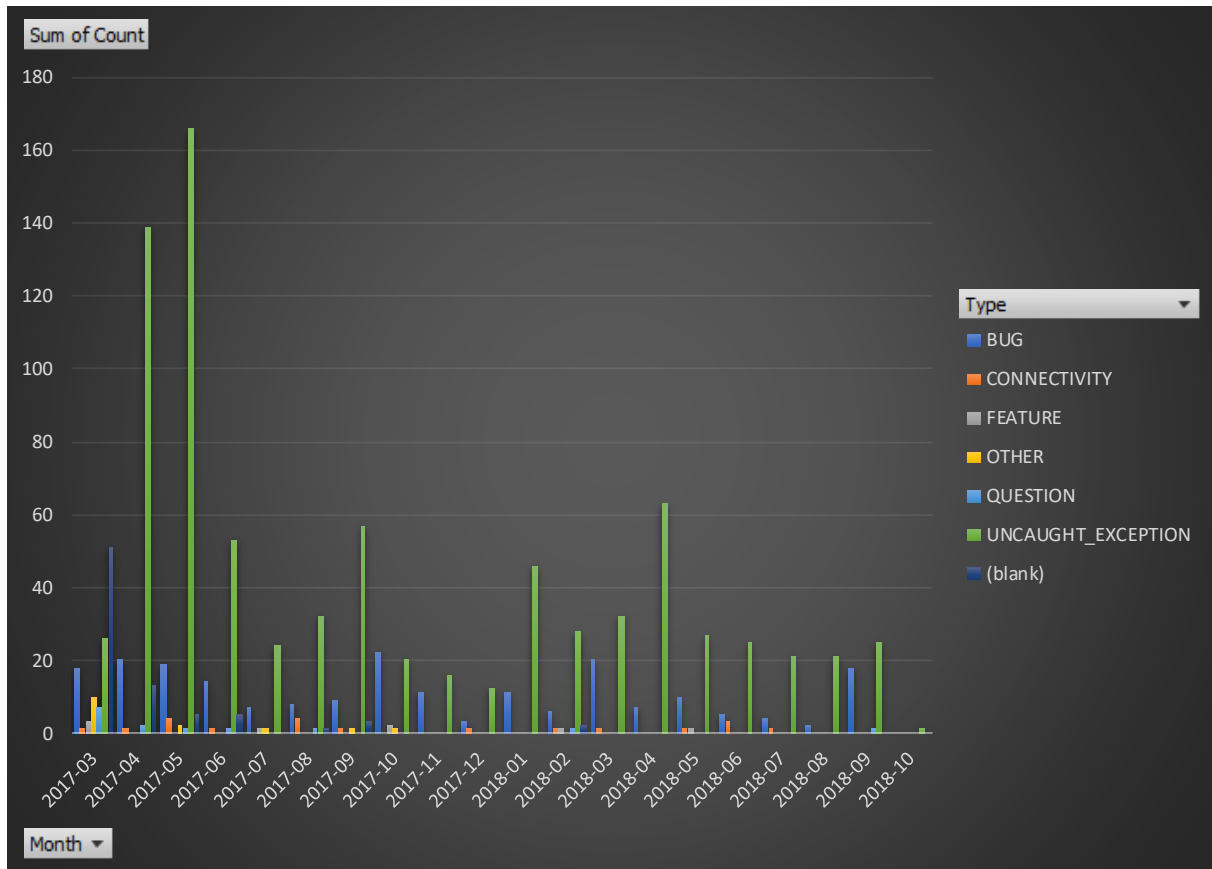


Figure 16 Support requests by type and over time

2.3.1 'connectivity' support requests

These requests are all related to connectivity issues between the experimenter and the testbeds. Main issues encountered were lack of IPv6-support, corporate firewalls blocking non-standard ports (Fed4FIRE+ testbed API's also use non-standard ports like 12369), unstable WiFi-connections.

Most of these connectivity-issues could be resolved by asking the user to enable the built-in proxy of jFed.

One support request showed a Man-in-the-middle attack being performed on the connections between the experimenter from Malaysia and the testbeds he was trying to use. Because of the security features in the Aggregate Manager API, these attacks caused the API-calls to fail.

2.3.2 'question_jfed' support requests

These requests were solved by explaining jFed functionality to the experimenter. When this information was lacking in the documentation, then it was added to clarify the issue for future users.

An example question was about the list of 'available nodes' which is available in the GUI. We needed to explain better that the information for populating this list was cached, which means

D2.2: Federation operations, tools and support

that it can be out-of-date for a few minutes, and doesn't reflect recently started/ended experiments.

2.3.3 'question_testbed' support requests

These requests were solved by explaining testbed functionality or error messages to the experimenter. Most of the error messages concerned insufficient available resources; the inability to extend an experiment because of future reservations and invalid configuration of a testbed resource request.

2.3.4 'feature_request' support requests

These requests concern feature requests for new functionality in the jFed GUI.

Sample feature requests are: better warning dialog layout, exposing more information about the reserved testbed resources to the experimenter or displaying the information in another format.

2.3.5 'bug_jfed' support requests

These requests concern unexpected behaviour of jFed due to bugs in the software.

These requests were then converted into a ticket in the jFed software issue tracker.

2.3.6 'bug_testbed' support requests

These requests concern unexpected behaviour of a testbed due to operational problems in the testbed.

These requests were then forwarded to the testbed support team to be resolved.

2.3.7 'bug_user' support requests

These requests were resolved by fixing configuration-errors in the environment in which jFed was run.

Most notably, errors of the user's PC-clock make that requests using timestamps fail because they are in the past/too far in the future.

Another error involved a Java-version which was incorrectly configured, which made the use of high-quality encryption mechanisms impossible

D2.2: Federation operations, tools and support

3 USABLE TESTBEDS

The two screenshots below show the number and locations of the testbeds that can be used with a Fed4FIRE account and the jFed tool.



Figure 17: Overview of available testbeds from the Federation monitor view

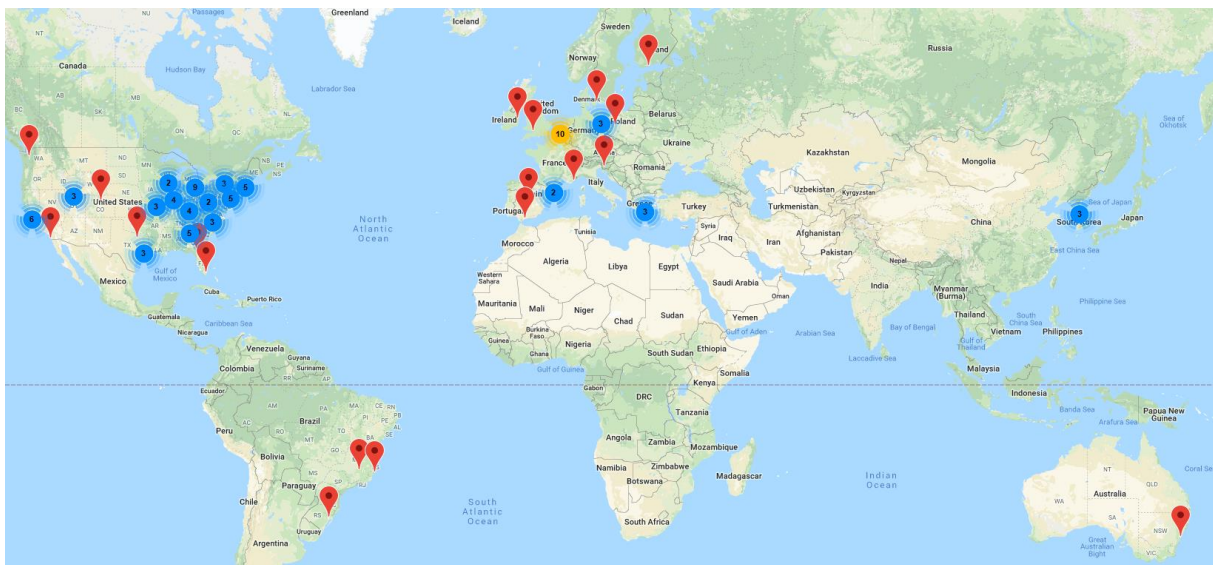


Figure 18: Overview of the available testbeds from the jFed view

4 OVERVIEW NEWEST JFED FEATURES

In this section we want to highlight a number of important new features that were added to jFed in Fed4FIRE+.

4.1 LINK TEST

When creating a network experiment, it is tedious to test manually all network links (as links can fail because of hardware problems, routing problems etc). Think e.g. about a class with 20 groups each using a 6 node experiment.

For this, a link test was added, running from jFed (so no testbed adaptations needed), testing all links in the experiment for connectivity and performance.

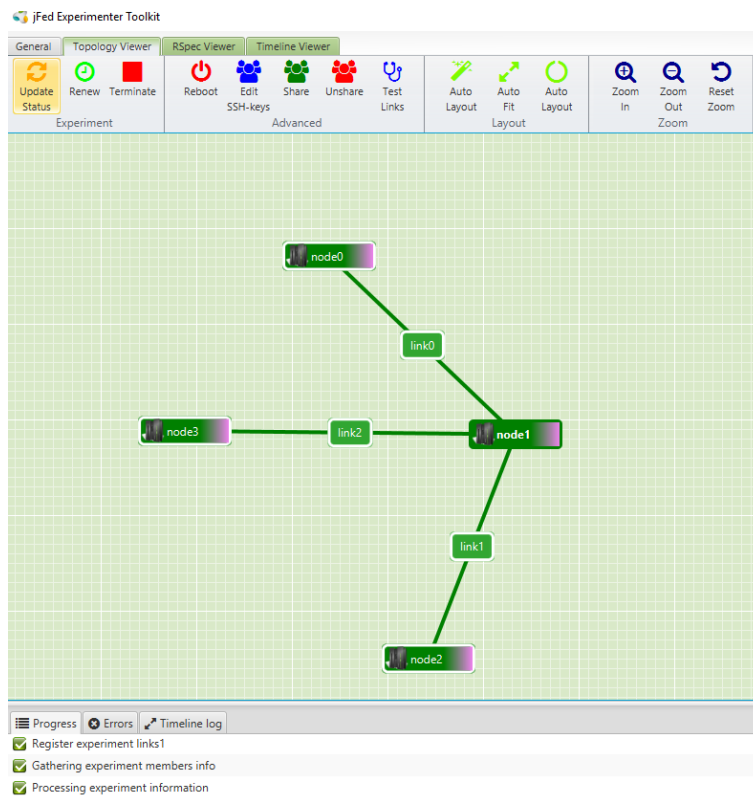
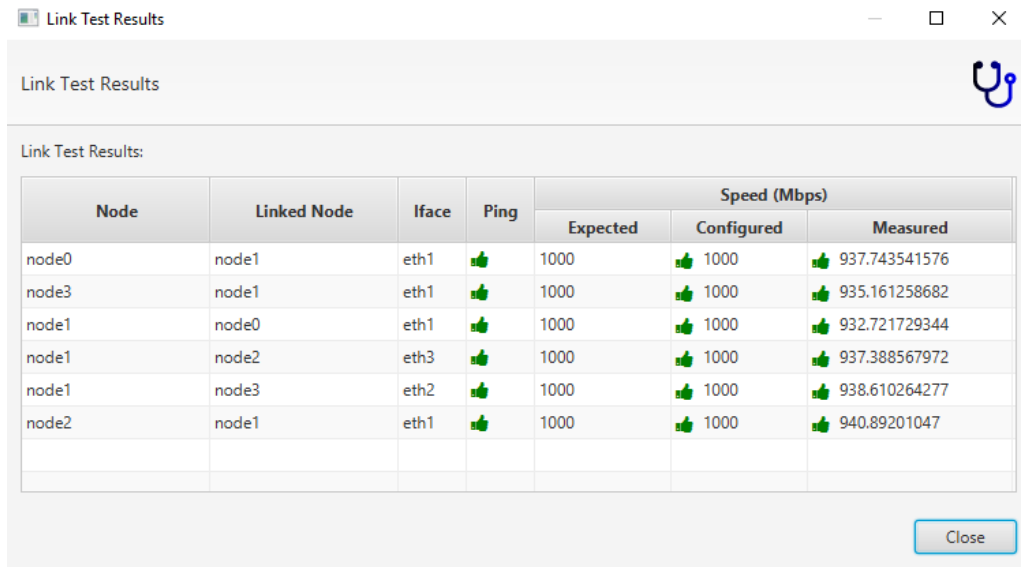


Figure 19: jFed network experiment for link testing (link test button can be seen on top middle)

D2.2: Federation operations, tools and support



The screenshot shows a window titled 'Link Test Results' with a table of test results. The table has columns for Node, Linked Node, Iface, Ping, and Speed (Mbps). The Speed (Mbps) column is further divided into Expected, Configured, and Measured. All ping tests show a green thumbs-up icon, indicating successful connections. The measured speeds are consistently high, ranging from approximately 932 to 941 Mbps.

Node	Linked Node	Iface	Ping	Speed (Mbps)		
				Expected	Configured	Measured
node0	node1	eth1	👍	1000	👍 1000	👍 937.743541576
node3	node1	eth1	👍	1000	👍 1000	👍 935.161258682
node1	node0	eth1	👍	1000	👍 1000	👍 932.721729344
node1	node2	eth3	👍	1000	👍 1000	👍 937.388567972
node1	node3	eth2	👍	1000	👍 1000	👍 938.610264277
node2	node1	eth1	👍	1000	👍 1000	👍 940.89201047

Figure 20: jFed link test results, including a speed test

4.2 CHOOSE A HEALTHY TESTBED FROM JFED

As described in Deliverable 3.2, there is a very extensive monitoring of the federation to make sure both experimenters and testbed owners are aware of possible problems and the availability of resources. The information of the monitoring is also directly visible in jFed.

As can be seen in the screenshot below, when choosing a specific resource type (e.g. bare metal/physical node), the experimenter can choose out of a list of testbeds, while having a clear view of planned maintenance (wrench icon), healthiness (color of the hearth icon) and availability of resources (bar diagram).

D2.2: Federation operations, tools and support

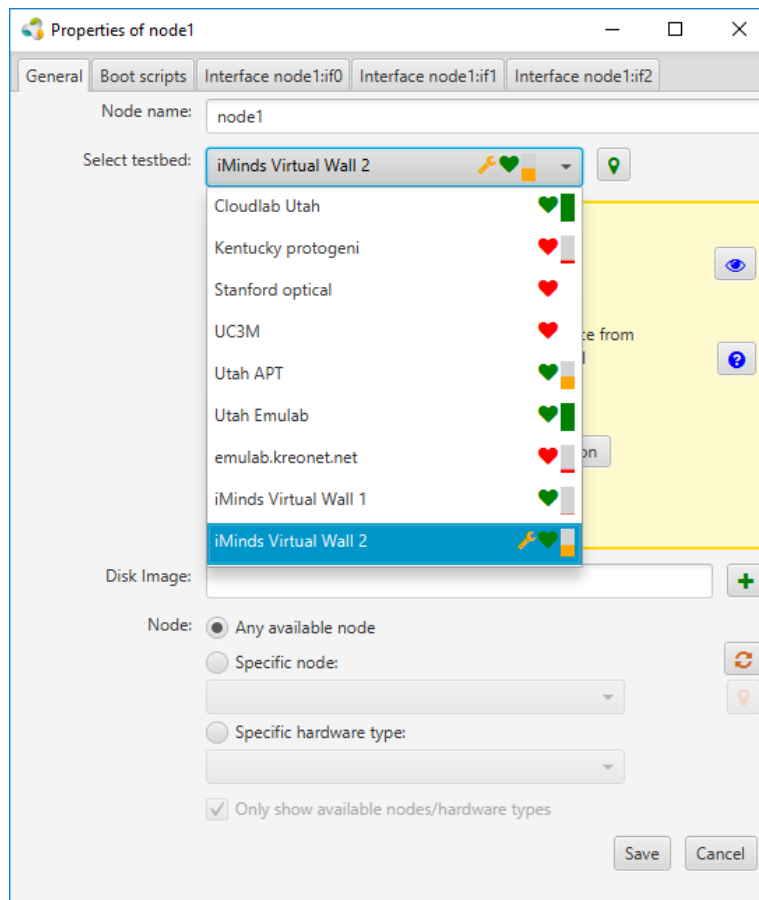


Figure 21: Testbed and resource availability visible in jFed

4.3 CHOOSE THE RIGHT HARDWARE FROM JFED

When choosing resources, sometimes it is needed to choose a specific type of resources. More info can now also be found directly from jFed. This information comes in via the Advertisement RSpec of the testbeds. Planned in the next cycle is to add also textual information on each hardware type (now this is documented in the documentation of the testbeds).

D2.2: Federation operations, tools and support

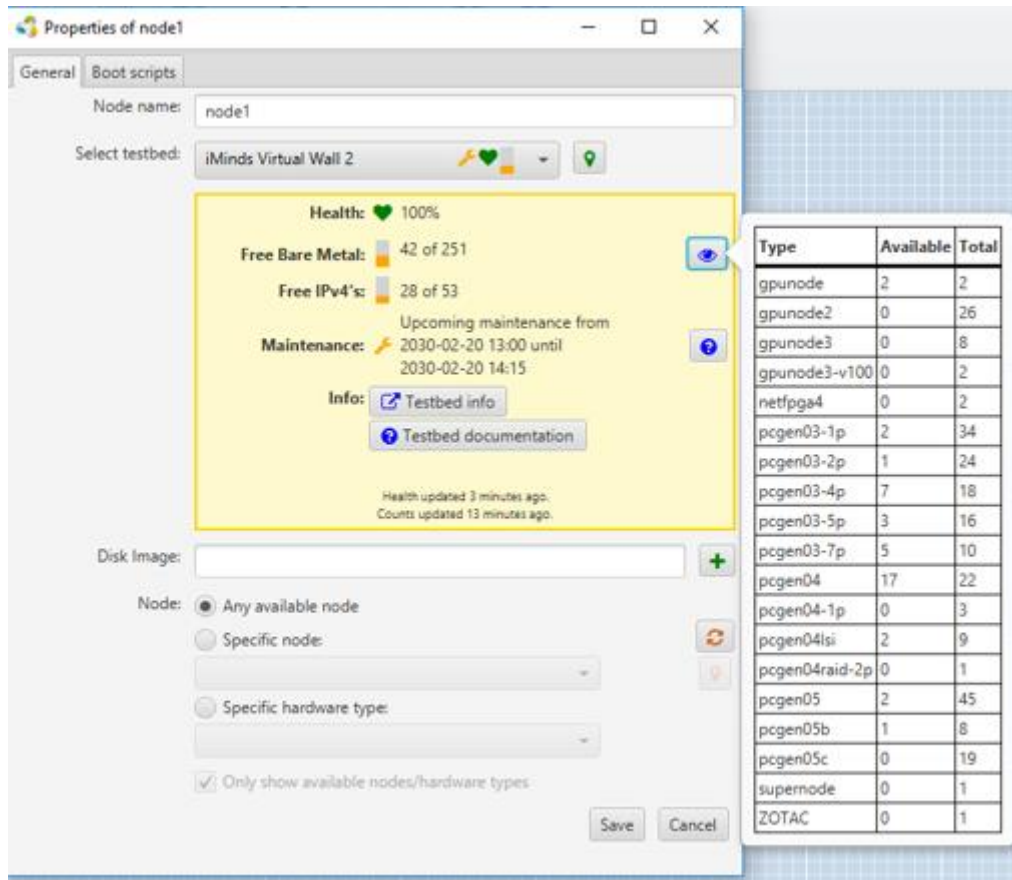


Figure 22: jFed availability per hardware type

4.4 GDPR SUPPORT

To support GDPR, we have foreseen a feature in jFed to have per testbed a specific webpage (hosted by the testbed) with their specifications and approval needs for GDPR. The details for the testbed side, are described in Deliverable 2.4. The screenshot below shows what happens when you start an experiment on a testbed needing such a specific terms and conditions.

D2.2: Federation operations, tools and support

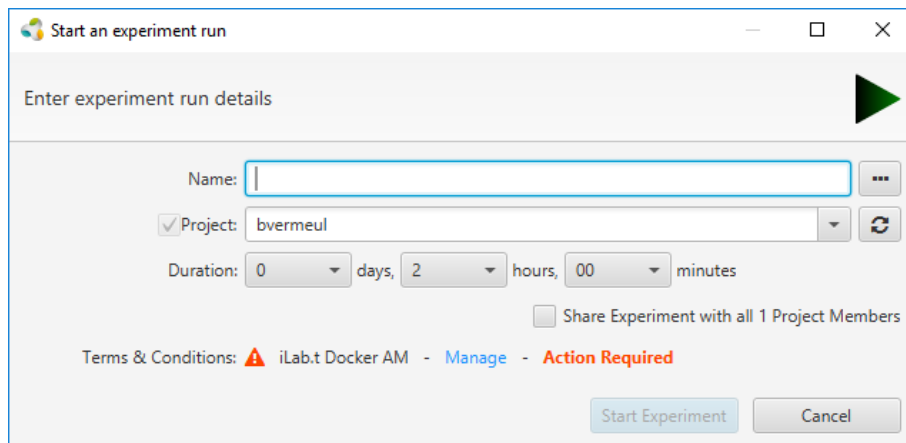


Figure 23: Starting an experiment with a testbed needing GDPR compliance check

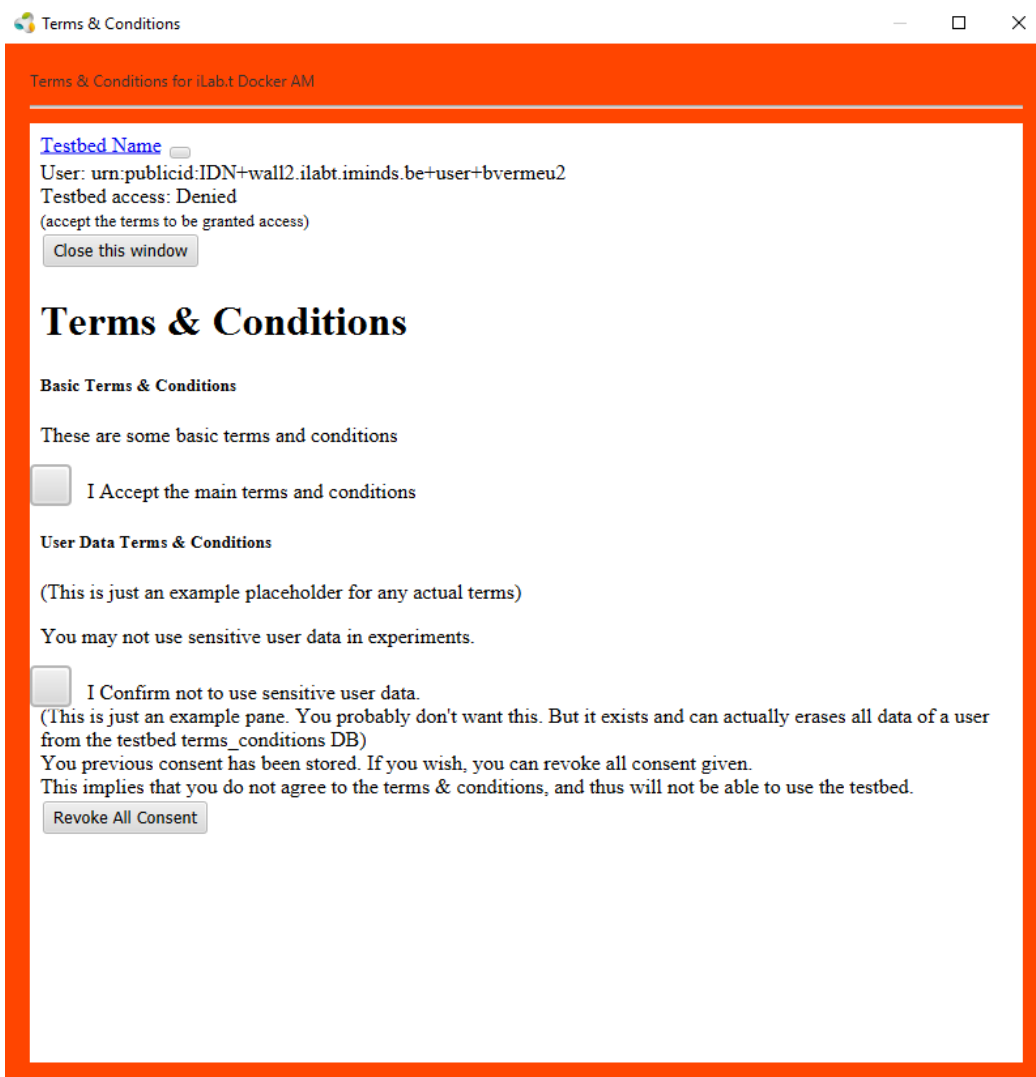


Figure 24: A specific (example) page per testbed for terms and conditions, opens in the built-in browser of jFed

D2.2: Federation operations, tools and support

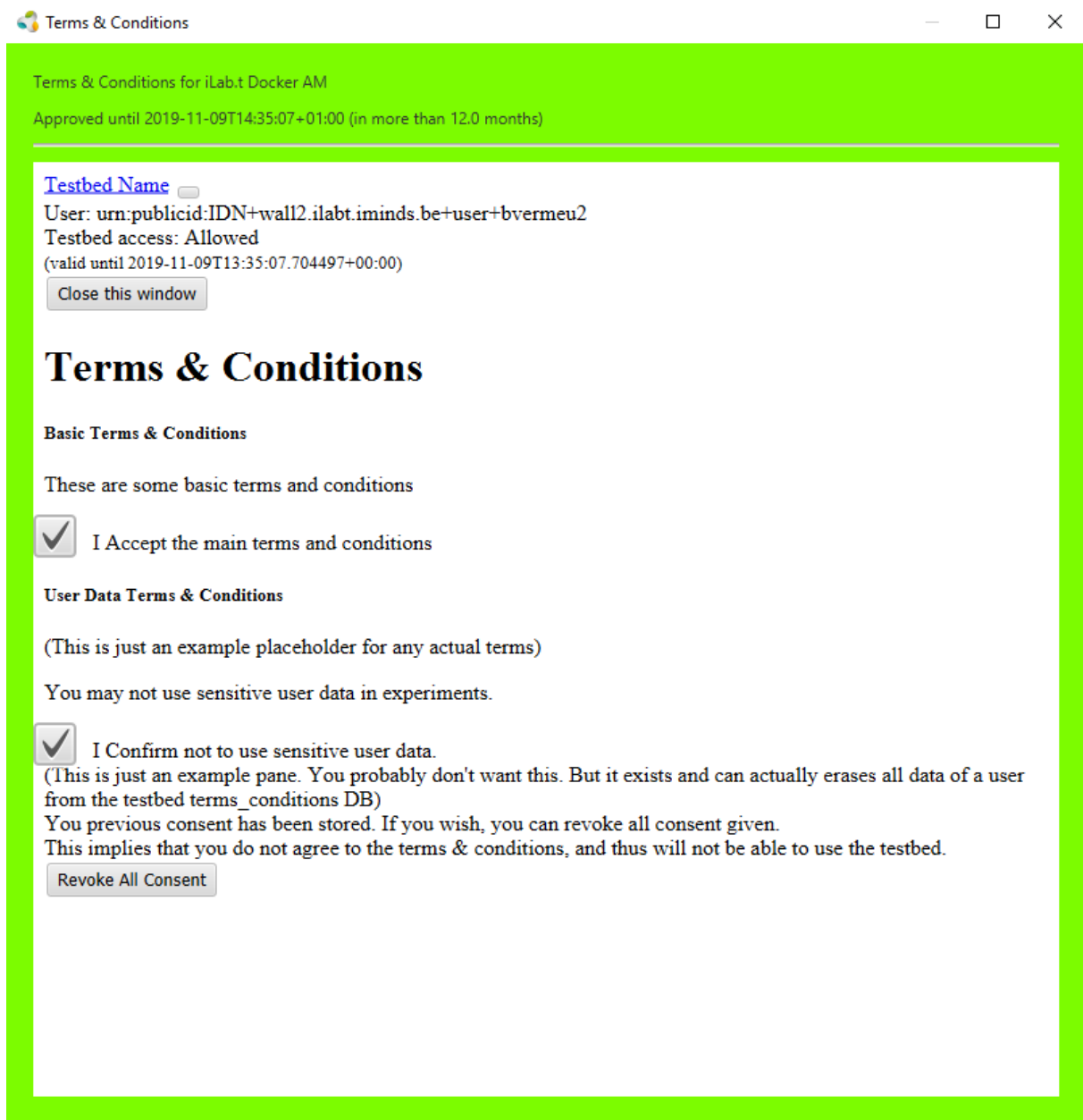


Figure 25: Same window, now after approval. Notice also the revoke button

After approval, next time you start an experiment, jFed remembers that you approved the terms and conditions for this testbed (it knows also when to ask them again, e.g. every month, every year. Each testbed can define this independently.):

D2.2: Federation operations, tools and support

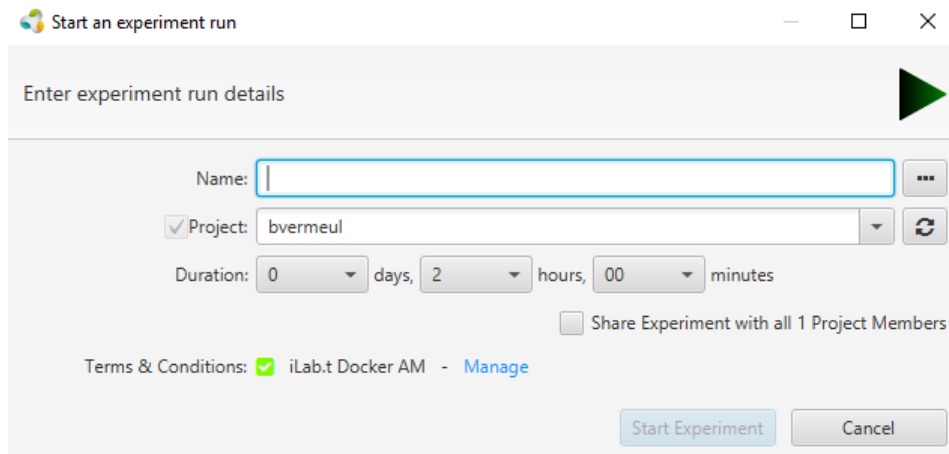


Figure 26: After approval of the terms and conditions

Of course, by clicking manage, you can again go to the same testbed page and revoke your consent.

4.5 FUTURE RESERVATION OF RESOURCES

Some of the testbeds do allow advance reservation of resources through a webinterface. This is now integrated in jFed with a built-in browser and automatic authentication on the website (through PKCS12 certificates).

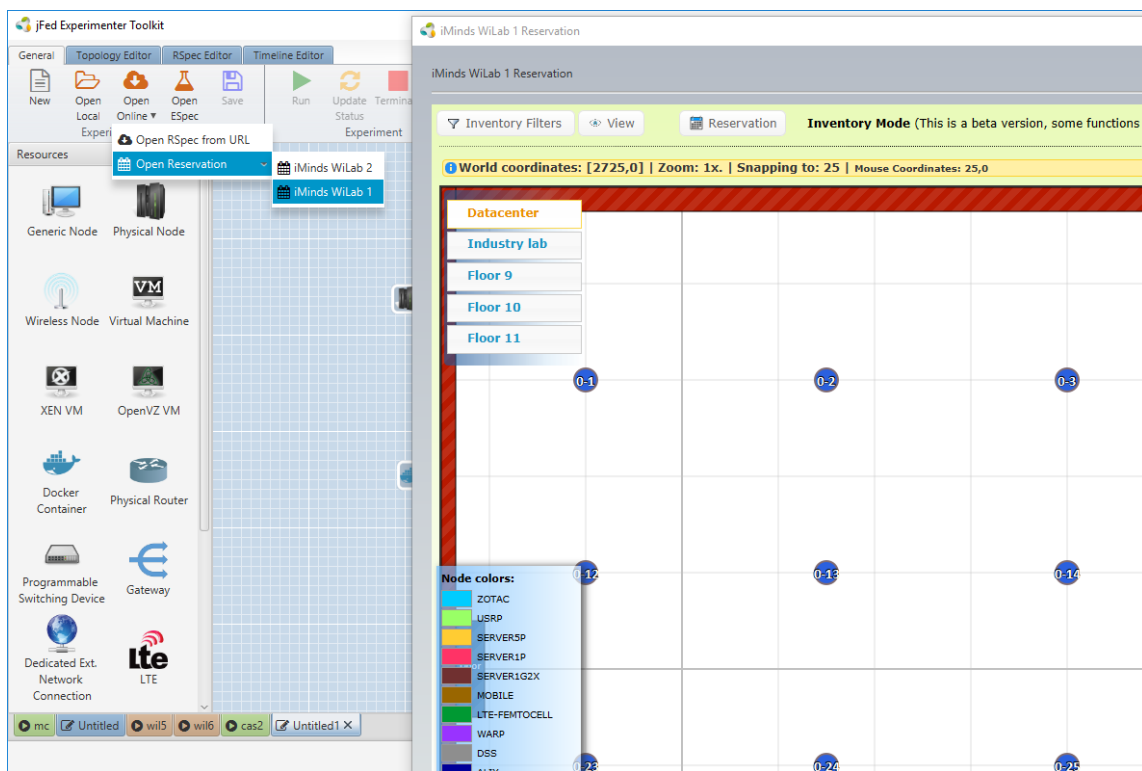


Figure 27: Opening a testbed advance reservation website from jFed

4.6 OTHER NEW FEATURES

Other features include:

- A completely rewritten command line version (CLI), see 6.4.
- Support for new testbeds and specific features (see deliverable D2.4).
- Experiment Specification for automated and reproducible experiments (see deliverable D3.2)
- To ease access to specific pages on nodes, when right clicking a node and choosing open browser, the experimenter can easily open a browser to a specific port on that node, see screenshot.

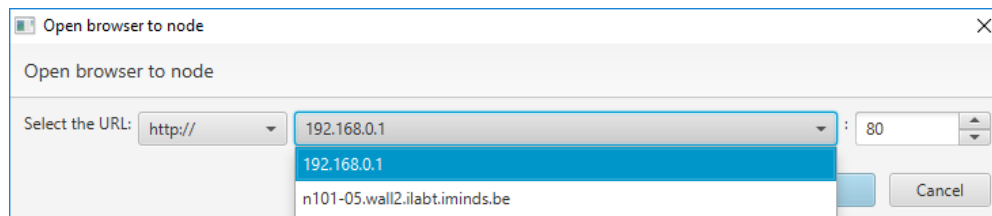


Figure 28: Open a webbrowser to a node, from jFed

5 APPENDIX: JFED FEATURES, USAGE AND USER MANUAL

This is the manual that appears online at <http://jfed.ilabt.imec.be>, more specifically at <https://doc.ilabt.imec.be/jfed-documentation/>. It highlights the features of jFed and how to use them. This is continuously updated (and versioned as can be seen at: <https://doc.ilabt.imec.be/jfed-documentation/>).

5.1 PREREQUISITES

5.1.1 Windows

You need to install [PuTTY](#).

You do not need to install java: it is bundled with the installer.

5.1.2 Linux

If you do not use the debian repository, you need to [install java 8 manually](#).

5.1.3 Mac

You do not need to install java: it is bundled with the installer.

But make sure that you start the jFed installer by downloading it, and then right-clicking and choosing 'Open'.

5.2 INSTALLATION

To install the jFed experiment GUI, go to the [jFed downloads page](#), and follow the instruction there.

5.3 BASIC FEATURES: FIRST EXPERIMENT TUTORIAL

5.3.1 Logging in

There are 2 basic methods to log into jFed:

Authority Login (recommended):

Either start jFed from the button on the site that provided your account, or start jFed and click on either "Login with Fed4Fire credentials" or "Login with GENI credentials".

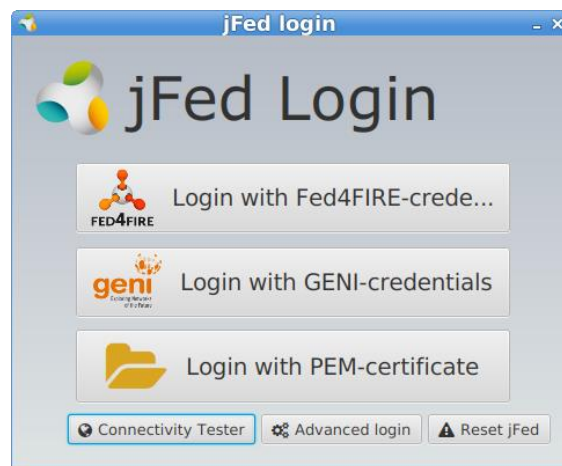
D2.2: Federation operations, tools and support

jFed might ask your account info and password. (After the first login, you'll only need your password)

Manual Login:

Alternatively you can start jFed, then click “Login with PEM-certificate”. You will need the “certificate file” (a PEM file containing your certificate and private key) that you can find at the website that provided your account.

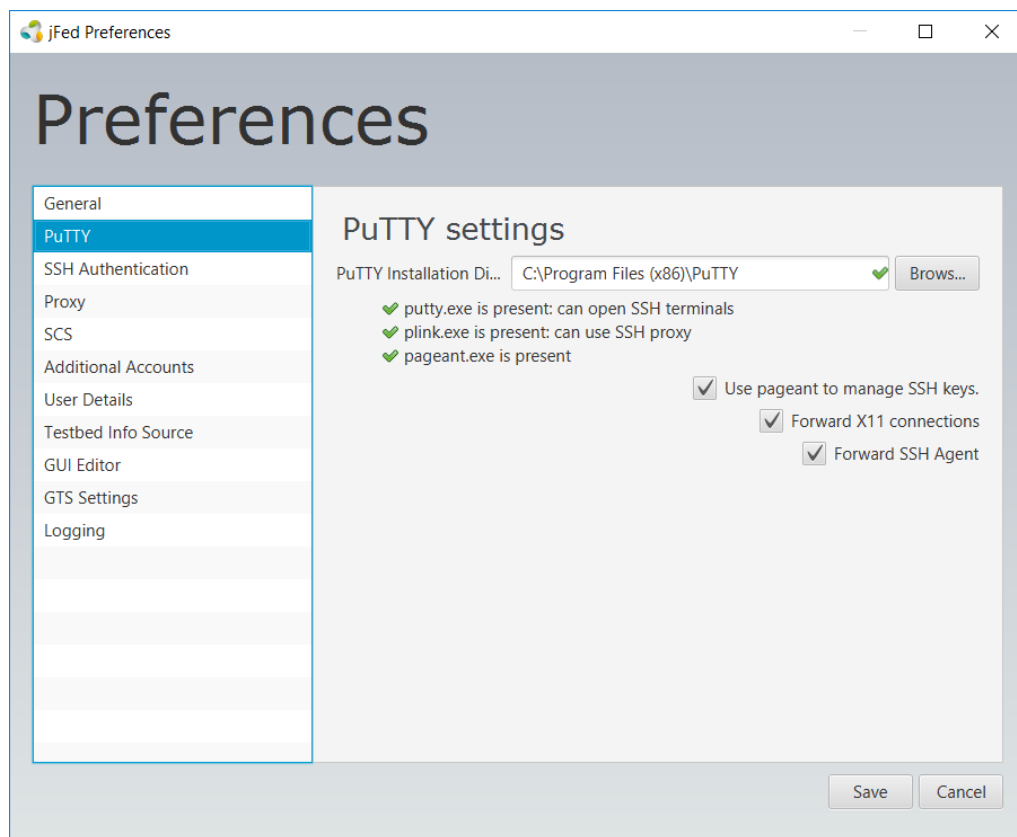
Then, on the next screen, open the “certificate file”, provide your password and press the login button.



After you login for the first time, a dialog box will pop up to say that you have to configure jFed for this initial run. You do typically do not need to make any changes, but it might be good to check out some settings.

D2.2: Federation operations, tools and support

5.3.1.1 Check Windows Preferences (optional)

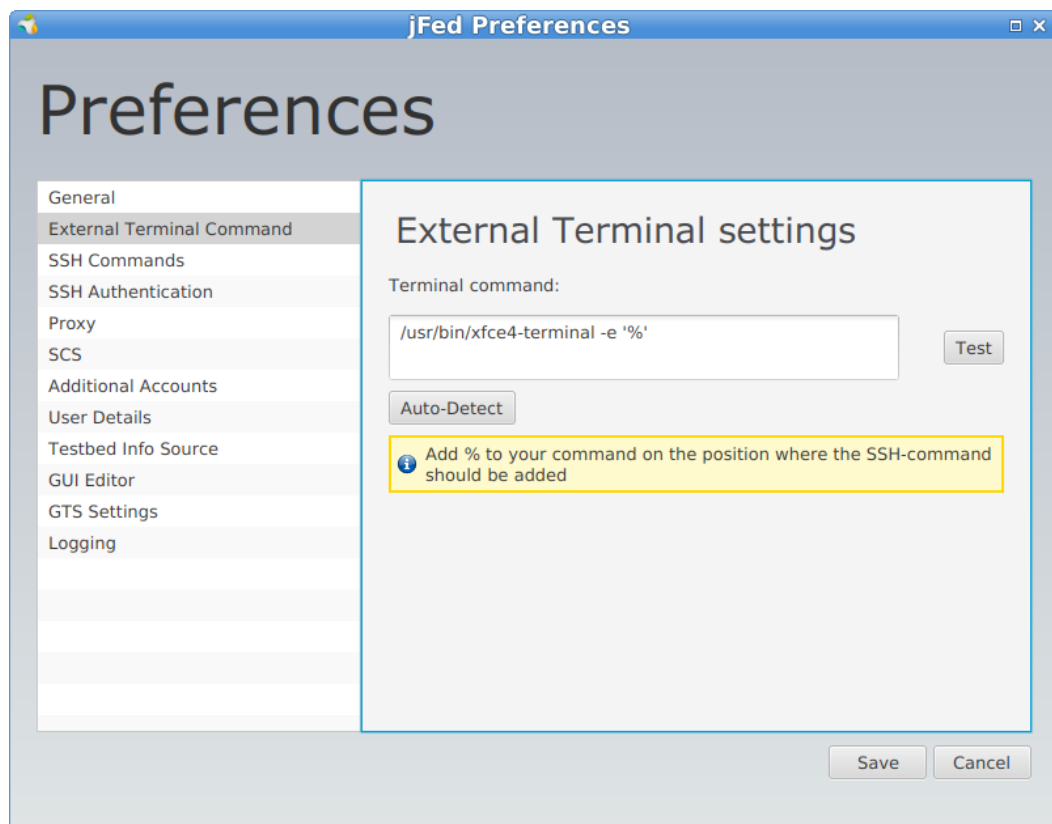


In this preferences settings you should point jFed to the PuTTY installation directory (see *putty*). You should see all green checkmarks. If not, then please install PuTTY from here: <http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.64-installer.exe>.

Click on `Use pageant to manage SSH keys` to enable the PuTTY ssh agent which makes that you only once have to type the passphrase on your ssh key. Click `Save` at the bottom right to save these settings.

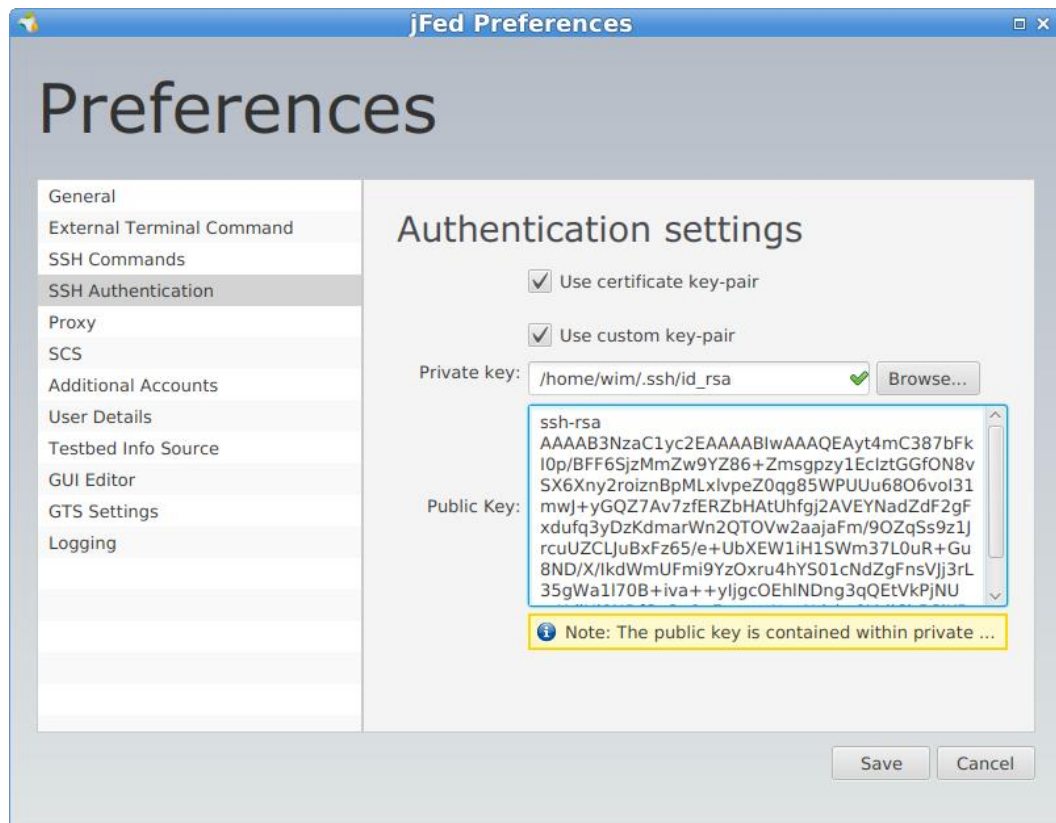
D2.2: Federation operations, tools and support

5.3.1.2 Check Unix/Mac Preferences (optional)



In this preferences settings, jFed should have a reasonable terminal configuration, so only change the default if it doesn't work when logging in.

D2.2: Federation operations, tools and support



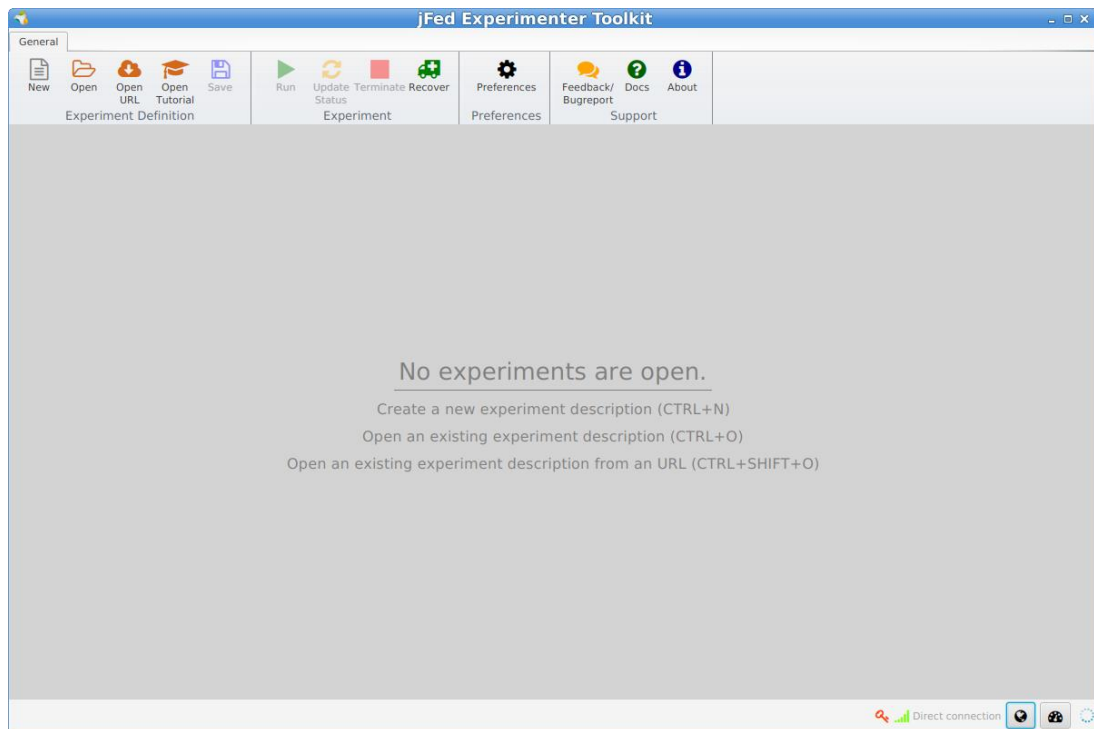
Secondly, you can click **Use custom key-pair** and point jFed to a *private and public SSH key* you have saved on your PC. This is not required, as your account is associated with an SSH key that jFed will use.

Click **Save** at the bottom right to save these settings.

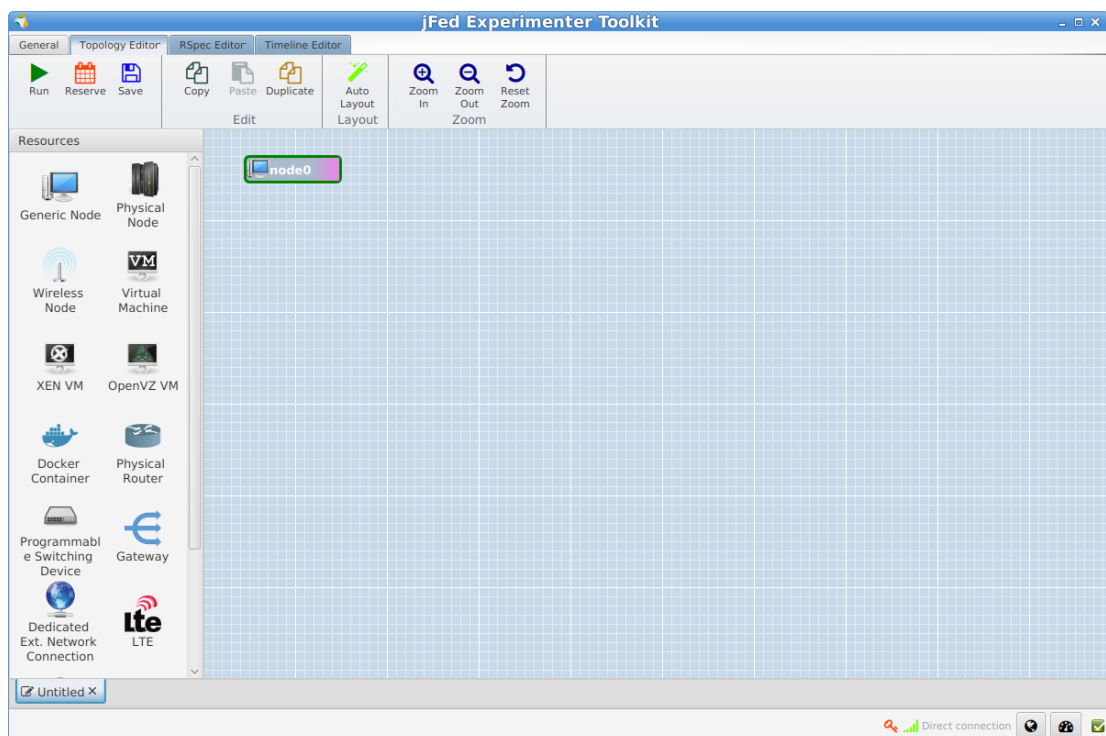
5.3.2 Create your first experiment

When you have logged in, and checked your preferences, you see jFed with no experiments loaded.

D2.2: Federation operations, tools and support

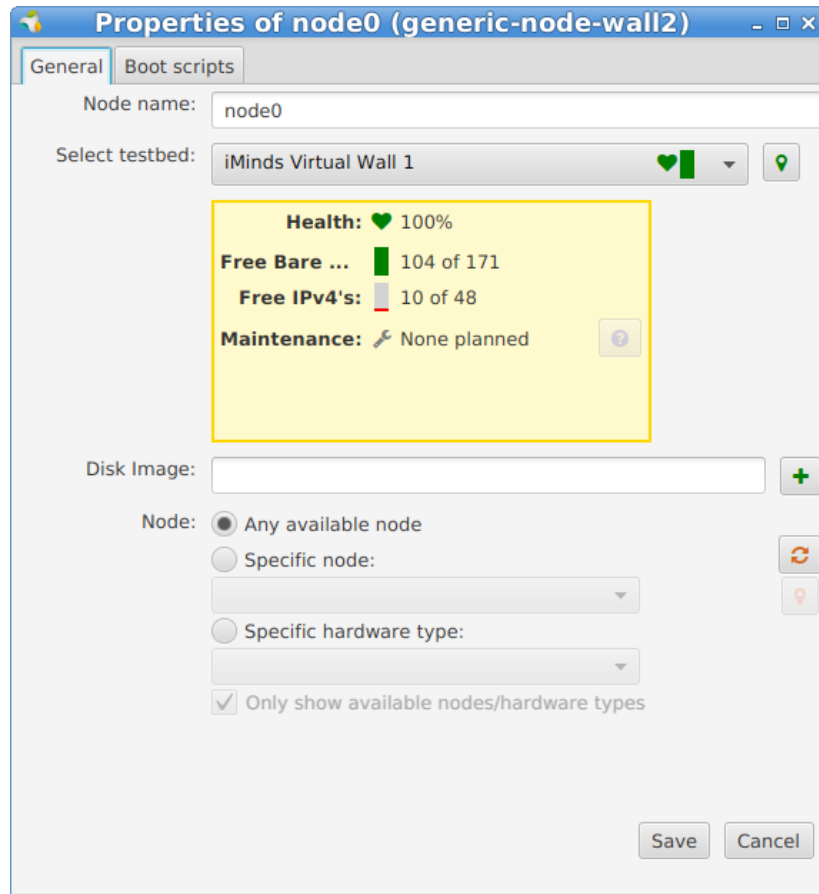


When you click **New**, you get a blank canvas where you can draw your experiment. Let's drag in a **Generic** node from the left side to the canvas.



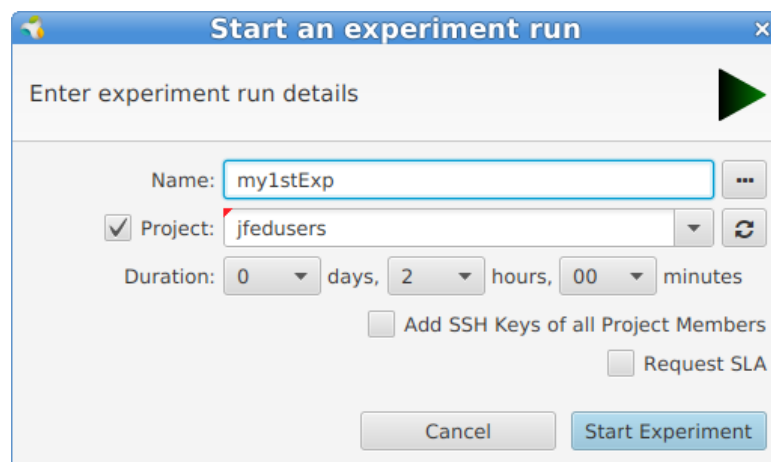
For more specific experiments you can right click and configure the node, but for now let it in the default settings.

D2.2: Federation operations, tools and support



5.3.3 Run the experiment

Let's run this experiment, by clicking the tab `General` at the top, and then the `Run` button. We will now have to choose a name for the experiment (= `slice name`) and choose a maximum duration.

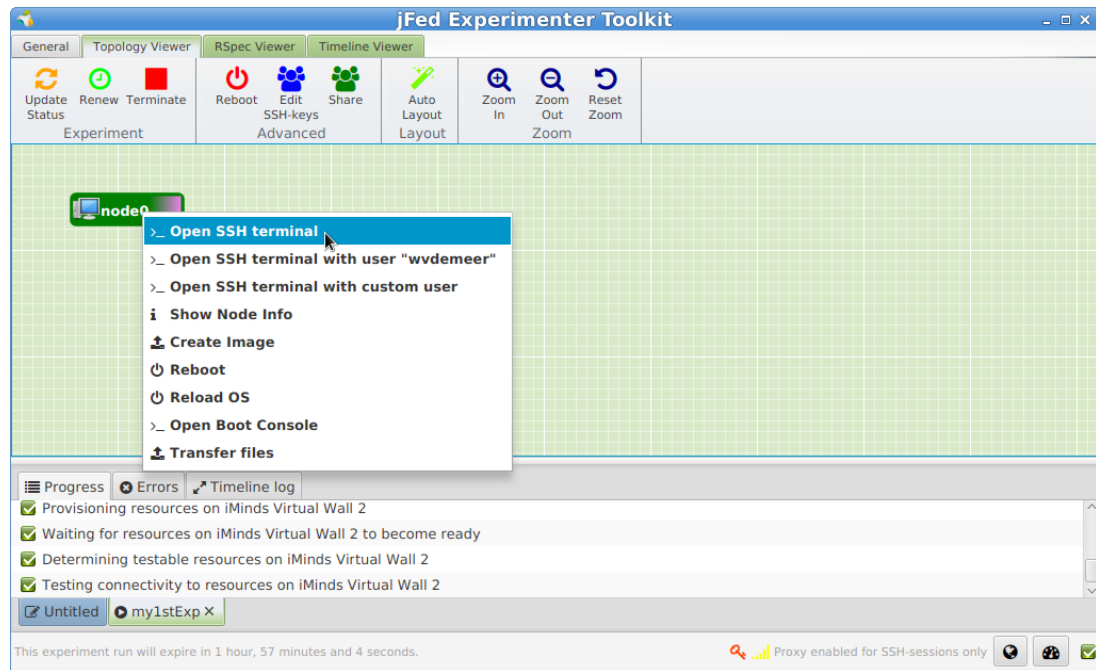


It will now take a couple of minutes to get the node prepared

D2.2: Federation operations, tools and support

5.3.4 Login on a node of the experiment

When the node becomes green, we can right click on the node, and click `Open SSH terminal`.

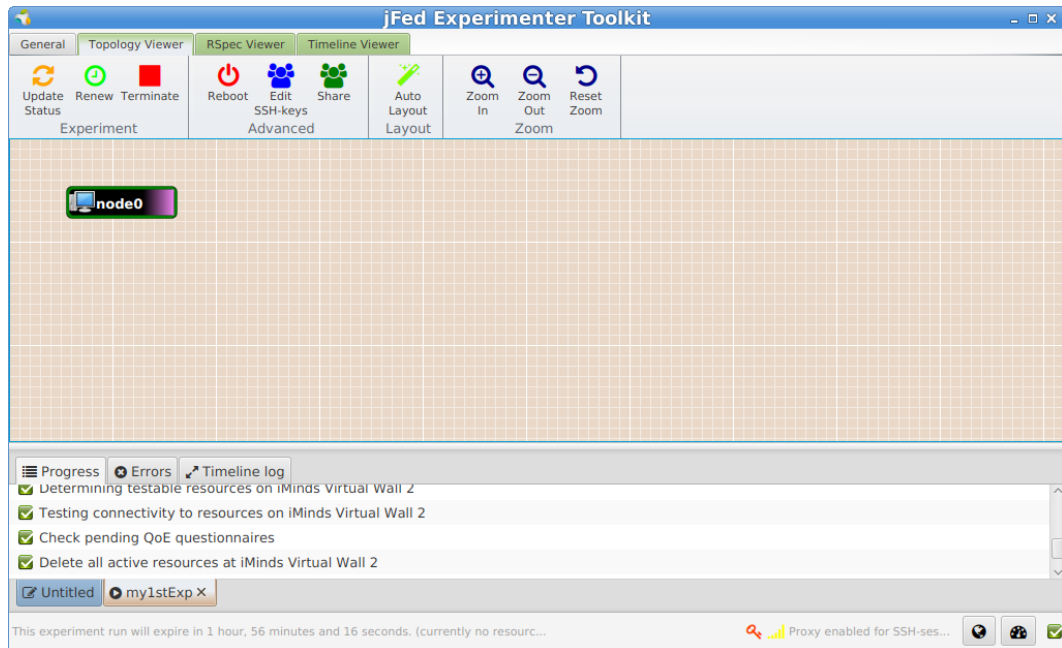


And then you should be automatically logged in. If the node says `Key refused` or another error it means something has gone wrong. See [Note on connectivity](#).

5.3.5 Ending the experiment

To release your resources before the end time of your experiment, you can click the `Terminate` button at the top in jFed. After that the nodes will become black and if your ssh connection is still open, you can see that the node will be shutdown.

D2.2: Federation operations, tools and support



5.3.6 Note on connectivity

As in Europe public IPv4 addresses are scarce, we have the following problems for getting connected to the nodes:

- Testbeds as Virtual Wall or w-iLab.t are only accessible through IPv6
- Some testbeds have only a limited number of public IPv4 addresses, which is minimal in relation to the number of virtual machines they run.
- Other testbeds assign only private IPv4 address to their nodes, and access is possible through a gateway node (with a single public IPv4 address).

We are currently working around this in several ways. For the scenario in this tutorial, this is how it works:

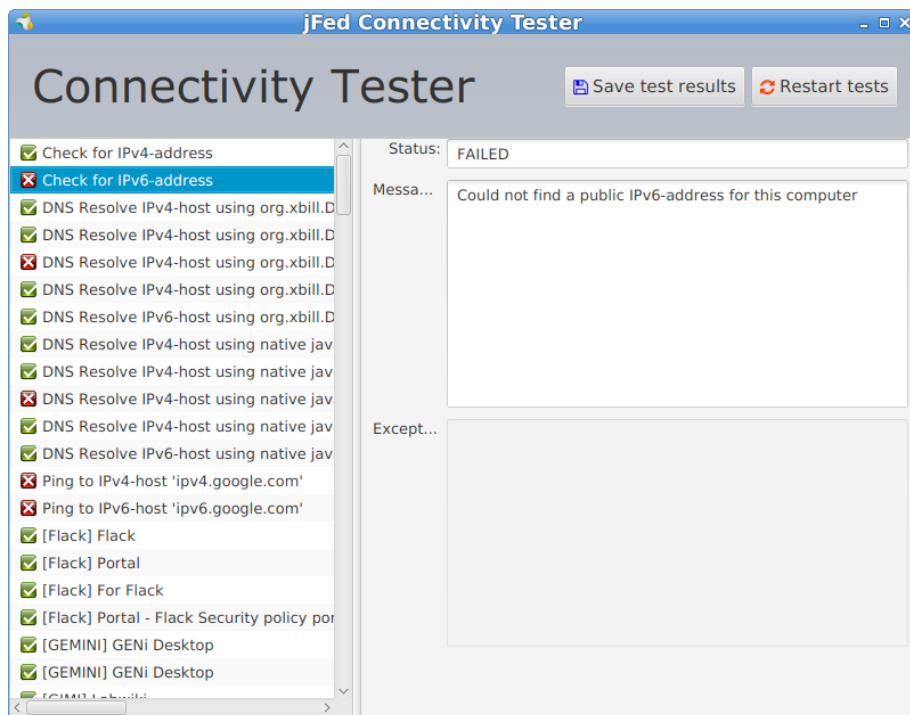
- The default node that was selected is at the Virtual Wall testbed (which is only accessible through IPv6).
- If you have IPv6 all will be okay and you will be able to login on the node.
- If you don't have IPv6, go to jFed preferences. Click **Run Proxy Test** at the bottom right, and then click **Always** next to **Proxy for SSH connections** followed by **Save**. You can now right click the node to login through SSH and you will be proxied through an IPv4 server.
- Alternatively, if you don't have IPv6, you can register for an IPv6 address and tunnel, e.g. at [Sixxs](#) (choose AYIYA tunnel) and install [Aiccu](#).

D2.2: Federation operations, tools and support

5.3.7 Test connectivity

You can test your connectivity with jFed by clicking the small *globe* button on the bottom.

And you will see a connectivity report:



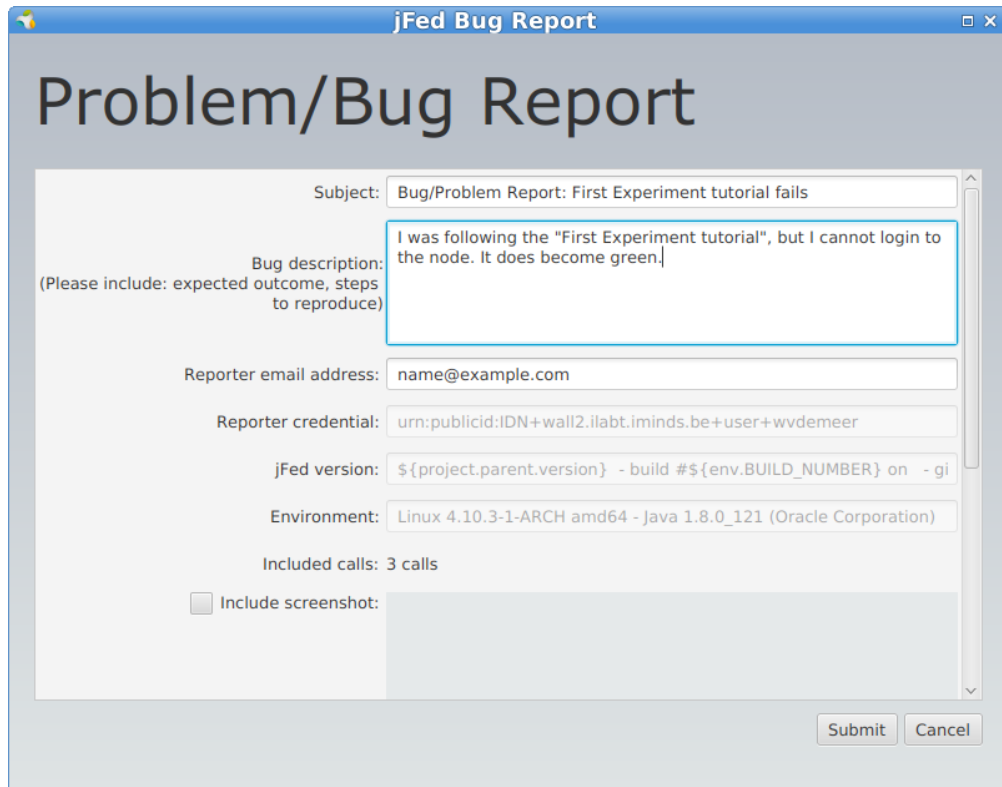
5.3.8 Feedback and Bug reports in jFed

In case you cannot get a green node on your canvas (e.g. at the bottom of jFed you see problems passing by), click the **Feedback / Bugreport** button in jFed and fill in a bug report. This will send all relevant information on calls and connectivity to jFed staff, so they can investigate the problem and report back to you.

The reporter email address is standard filled in with your certificate email address and this is forwarded by the authority to your own email address. You don't have to change this, but you can if you like.

Besides bug reports, you can also send question, feature requests, ...

D2.2: Federation operations, tools and support



The screenshot shows a web browser window titled "jFed Bug Report" with a "Problem/Bug Report" form. The form contains the following fields and content:

- Subject:** Bug/Problem Report: First Experiment tutorial fails
- Bug description:** (Please include: expected outcome, steps to reproduce) I was following the "First Experiment tutorial", but I cannot login to the node. It does become green.
- Reporter email address:** name@example.com
- Reporter credential:** urn:publicid:IDN+wall2.ilabt.iminds.be+user+wvdemeer
- jFed version:** \${project.parent.version} - build #\${env.BUILD_NUMBER} on - gi
- Environment:** Linux 4.10.3-1-ARCH amd64 - Java 1.8.0_121 (Oracle Corporation)
- Included calls:** 3 calls
- Include screenshot:**

At the bottom right of the form are "Submit" and "Cancel" buttons.

5.4 ADVANCED FEATURES

5.4.1 Introduction

This section will show you some advanced features of the jFed Experimenter GUI.

5.4.2 Proxy settings

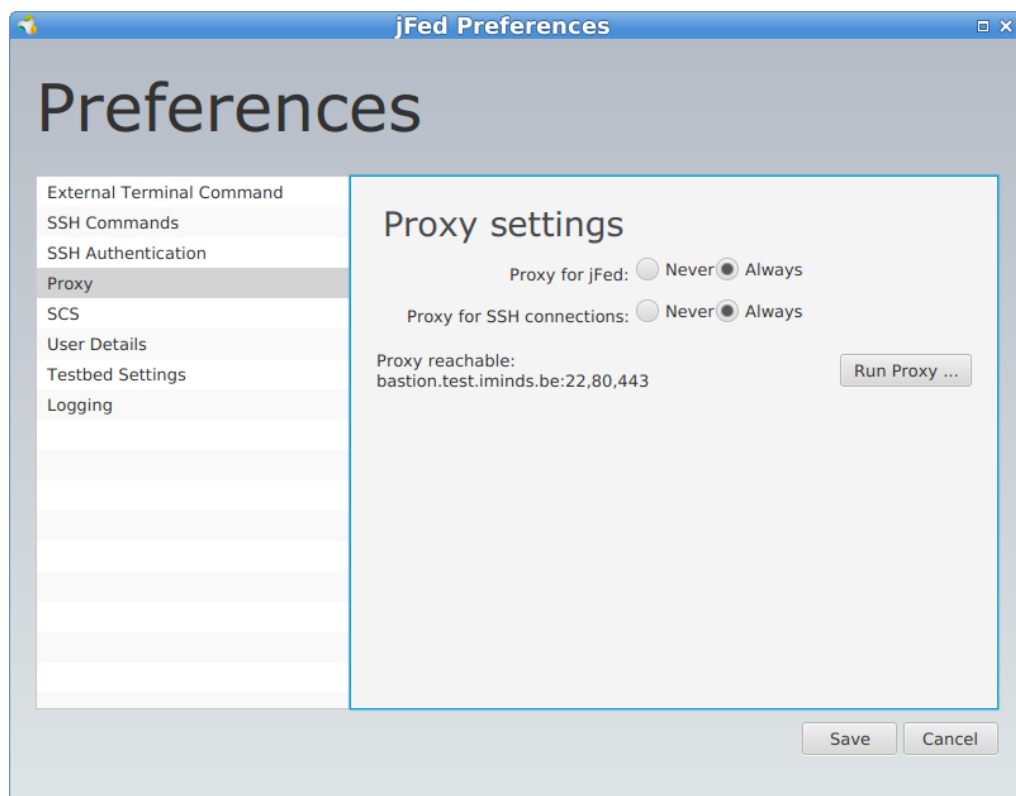
One of the advanced features of jFed is the support of an SSH proxy for the following three problems:

- the APIs of testbeds typically use a lot of exotic TCP ports (12369, 12345, 3636, ...). These ports are sometimes blocked by firewalls.
- some testbeds (e.g. Virtual Wall and w-iLab.t) expose the nodes only through IPv6 (because of shortage of public IPv4 addresses). If you are only on an IPv4 network, you cannot reach those nodes without an IPv6 tunnel.
- some nodes (e.g. Virtual Wall windows 7 images, Virtual Wall OpenVZ containers) might have only a private IPv4 address and be accessible only through a gateway (with a public IP address).

D2.2: Federation operations, tools and support

For all these cases, jFed has now SSH proxy support which can be enabled for the API calls or/and SSH connections. This can be enabled by going to the “General” tab, clicking “Preferences” and going to the “Proxy” settings. The settings are labeled respectively `Use Proxy for jFed` and `Proxy for SSH connections` in the screenshot below. Before you can tick `Always` you should click first `Run Proxy Test`. If you cannot reach the proxy server, it makes no sense to use the proxy.

On windows, it is good to combine this with using the PuTTY SSH agent, this can also be set in the preferences.



5.4.3 Recover slices

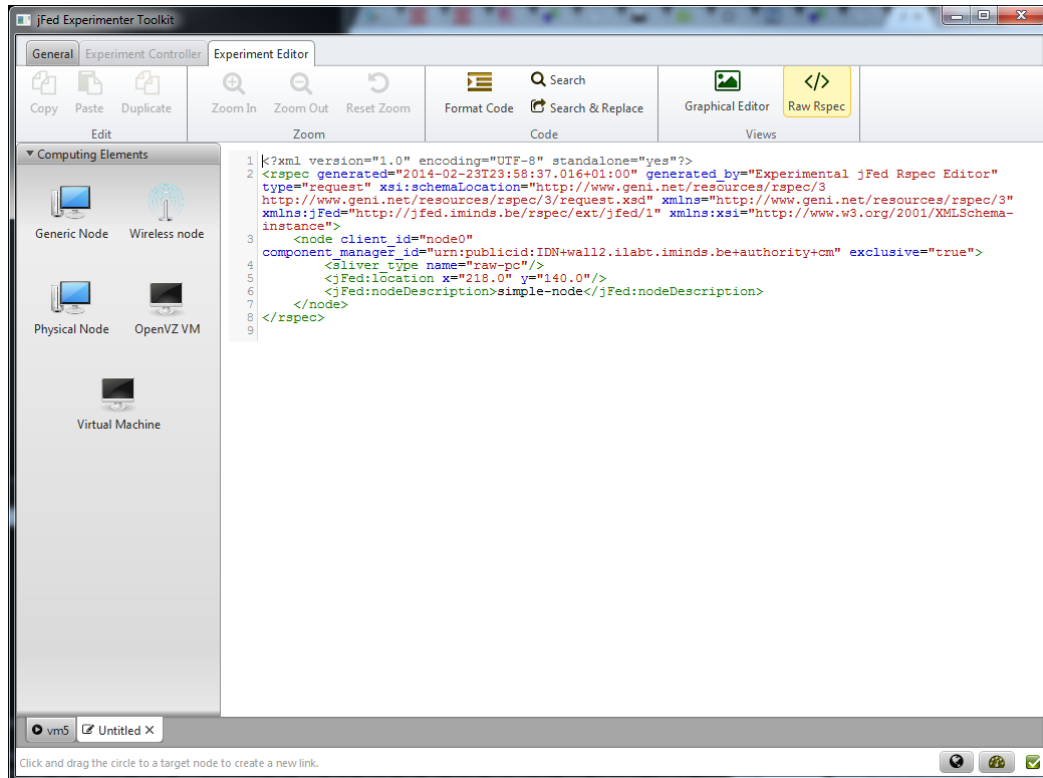
If you have set up an experiment/slice, and you have terminated jFed, you can click the `Recover` button to fetch the information of old slices. In this way you can again see the topology and login on the nodes if they are still up and running.

5.4.4 RSpec editor

It is now possible to toggle between a graphical editor and a raw RSpec editor, which makes it possible to add extra information or to put in new RSpecs. With

D2.2: Federation operations, tools and support

the **Format Code** button you can optimise the XML view on the RSpec. With **Search** or **Search & Replace** you can enhance your edit. E.g. if you change all `component_manager_id`'s from `wall2.ilabt.iminds.be` to `emulab.net`, your same experiment will run in Utah Emulab.



As we have now a raw RSpec editor, it has become possible to support RSpecs which are difficult to view graphically, e.g. openflow.

If you **Open URL** and fill in <http://jfed.iminds.be/openflow.rspec>, an openflow RSpec is loaded for the Virtual Wall openflow testbed and you can further edit it. jFed supports sending the RSpec to the right Aggregate Manager.

See also <http://fed4fire-testbeds.ilabt.iminds.be/ilabt-documentation/openflow.html>.

5.4.6 Virtual wall advanced features

You can find the specific Virtual Wall advanced RSpecs at [Virtual Wall RSpecs](#) which make it possible to load other images, Windows 7, impairment on links, install software, and so on.

D2.2: Federation operations, tools and support

5.4.7 Adding extra ssh-keys to nodes

By default jFed will add the ssh-key that you specify in preferences, and the public ssh key extracted from your pem certificate. However if you want to specify extra ssh-keys (e.g. of other users, or from yourself but on another device), you can do so as follows, by adding the following information in the RSpec (just before the closing `rspec` tag e.g.):

```
<jfed-ssh-keys:user-ssh-keys
user="urn:publicid:IDN+wall2.ilabt.iminds.be+user+wvdemeer">

  <jfed-ssh-keys:sshkey>ssh-rsa AAAAB... wim@tzu</jfed-ssh-keys:sshkey>

</jfed-ssh-keys:user-ssh-keys>
```

You can repeat both the internal `sshkey` (this will add multiple ssh-keys to the specified user) or the external `user-ssh-keys` so you can specify multiple users.

If you omit the `user` part, then the ssh-keys will be added to the default user who created the experiment. The user unique identifier (URN) depends on the authority you use, but for users of the Virtual Wall 2 authority, this always starts with `urn:publicid:IDN+wall2.ilabt.iminds.be+user` followed by your login name.

Caveats:

- this is a jFed specific extension to the RSpec (it is interpreted and added as argument in the AM calls), so specify in the rpec header (jFed itself does this already for new experiments, but not if you import an old RSpec):

```
xmlns:jfed-ssh-keys="http://jfed.iminds.be/rspec/ext/jfed-ssh-keys/1"
```

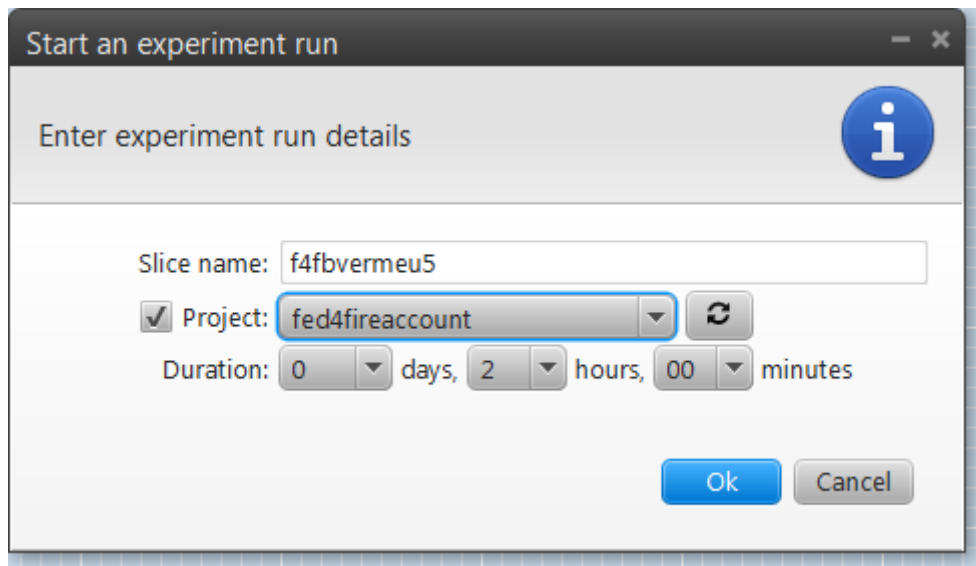
- you need a jFed compile r1138 or newer (<http://jfed.iminds.be/releases/r1338/>: click Experimenter GUI - Launch webstart)



D2.2: Federation operations, tools and support

5.4.8 Support for subauthorities

When starting an experiment, automatically a list of your subauthorities/projects is shown and you can select one. This information can be used at testbeds to attach policies or quota based policies.



5.4.9 Ansible support

jFed supports [ansible](#) in two ways: either run ansible playbooks on your local PC (requires linux/MAC), or add a node to your experiment and run ansible from that node.

5.4.9.1 Local ansible support

5.4.9.1.1 *Generate ansible inventory file from running experiment*

You can generate an ansible [inventory file](#) from any running experiment. To do this, open the “Rspec Viewer” ribbon. Then click the “Save Experiment” button and select “Export Configuration Management Settings”. You then need to choose a location to store a zip file. The generated zip file contains ansible.cfg and the ansible inventory file. Note that in some cases, this file might contain your private key! So it is not a good idea to share this file.

5.4.9.1.2 *Automatically run local ansible*

Prerequisites: This will only work when ansible is locally installed (on linux/mac).

D2.2: Federation operations, tools and support

It is also possible to make jFed execute ansible playbooks, using the local ansible install, when the experiment is ready. To do this, add the `execute_ansible_playbook` element to the rspec:

```
<rspec ...>

...

  <jFed:execute_ansible_playbook xmlns:jFed="http://jfed.iminds.be/rspec/ext/jfed/1"
source="https://example.com/playbook.yml" output_file="output.txt"/>

</rspec>
```

The source can be the path to a local file as well.

5.4.9.2 Remote ansible: Ansible node

Prerequisites: This feature expects Ansible to be installed on one of the nodes in the experiment, which can be achieved by using an install-script, or by embedding it in a custom disk-image.

jFed supports the automatic execution of Ansible playbooks on the ansible node in your experiment. This feature is included in jFed v5.7 and higher. These playbooks are launched when your experiment starts. The output of the playbook can be inspected in an extra “Ansible”-tab that will appear during the execution.

5.4.9.2.1 Feature: Waiting for install-scripts to complete

Some testbeds (like Virtual Wall 1&2, and other emulab-testbeds) have support for the automatic execution of install-scripts once a node has been provisioned. However, there is no feedback on when these scripts have finished. To prevent race-conditions (eg. trying to execute the ansible playbook before ansible has been installed on the host machine), an extra attribute `jfed:finished_flag` has been added to the execute-tag:

```
<services>

  <install install_path="/local" url="http://doc.ilabt.iminds.be/jfed-documentation-5.7/_static/install-ansible.tar.gz" />
```


D2.2: Federation operations, tools and support

```
<execute shell="sh" command="cd /local && sudo /bin/bash install-ansible.sh"

  jfed:finished_flag="/tmp/ansible-install-finished"/>

</services>
```

This attribute must contain the location of a file that will be created (`touch-ed`) at the end of the script. This way, jFed knows the script has finished executing, and can continue the provisioning process. An example of a script installing ansible **can be found here**.

5.4.9.2.2 Feature: Automatic distribution of a shared private key

As the Ansible-playbook is run from one of the nodes in the experiment, this node must be able to login on all the nodes. By adding an `<jfed:distribute_ssh_keypair>`-tag, jFed will automatically create and distribute a public/private keypair to allow all nodes to login on each other.

```
<jfed:distribute_ssh_keypair />
```

This tag has the following optional arguments:

- `user`: the user for which a keypair must be generated. This defaults to the user starting the experiment
- `location`: the path where the private key must be stored. This defaults to `~/.ssh/id_rsa`.

In practice, it is almost never necessary to use these arguments.

5.4.9.2.3 Feature: Dynamic inventory generation

jFed can automatically generate an inventory file. To allow differentiation between nodes, you can use `<jfed:ansible_group>`-tags to add a node to one or more groups in the inventory.

For example:

D2.2: Federation operations, tools and support

```
<node client_id="serverA" exclusive="true"
component_manager_id="urn:publicid:IDN+wall2.ilabt.iminds.be+authority+cm">

  <sliver_type name="raw-pc"/>

  <jfed:ansible_group name="server"/>

</node>
```

Will result in:

```
[server]

serverA ansible_host=n063-17a.wall2.ilabt.iminds.be
```

This inventory file will be uploaded to the location specified in `<jfed:ansible inventory="/MY/LOCATION">`.

Feature: Executing an Ansible playbook
~~~~~ To request the execution of an Ansible playbook, you must add a `<jfed:ansible>`-tag to the `<services>`-tag of a node.

This tag has the following arguments:

- **galaxy-command:** The command necessary to execute Ansible Galaxy. Defaults to `ansible-galaxy`
- **install\_requirements:** A absolute path of a file containing all the ansible 'roles' to be installed by Ansible Galaxy
- **inventory:** The absolute path where jFed must upload the dynamically created files to
- **playbook-command:** The command necessary to execute Ansible Playbook. Defaults to `ansible-playbook`.
- **execute\_playbook:** The absolute path to the playbook that must be executed by Ansible Playbook
- **debug:** request extra verbose output from Ansible (equivalent to executing ansible with `-v`)

## D2.2: Federation operations, tools and support

For example:

```
<jfed:ansible

    galaxy-command="sudo ansible-galaxy"

    install_requirements="/local/my_repo/playbooks/ansible-requirements.yml"

    inventory="/local/my_repo/playbooks/hosts"

    playbook-command="sudo ansible-playbook"

    execute_playbook="/local/my_repo/playbooks/main.yml"

    debug="true"

/>
```

### 5.4.10 Experiment Specification (espec)

jFed 5.8 (and higher) support the Experiment Specification format, or espec. This allows you to create a “bundle” containing multiple files that define an experiment. This bundles typically contains an RSpec and scripts and files to be uploaded to the resources.

More information at the [jFed Experiment Specification page](#).

## 5.5 SHORTCUTS

### 5.5.1 General shortcuts

- **F1** Open documentation
- **F2** Report a Bug
- **F12** Preferences

### 5.5.2 Experiment definition

- **CTRL+N** New experiment definition

## D2.2: Federation operations, tools and support

- **CTRL+O** Open experiment definition from file
- **CTRL+SHIFT+O** Open experiment definition from URL
- **CTRL+S** Save experiment definition
- **CTRL+P** Run experiment

### 5.5.3 Topology editor

- **CTRL+C** Copy selected element
- **CTRL+V** Paste
- **CTRL+D** Duplicate selected element
- **CTRL+L** Auto Layout
- **CTRL++** Zoom in
- **CTRL+-** Zoom out
- **CTRL+0** Reset zoom

### 5.5.4 RSpec editor

- **CTRL+ALT+L** Format RSpec
- **CTRL+ALT+V** Verify RSpec
- **CTRL+F** Search
- **CTRL+SHIFT+F** Search & Replace

### 5.5.5 Timeline editor

- **ALT+INSERT** Add command
- **CTRL+B** Add barrier
- **CTRL++** Zoom in
- **CTRL+-** Zoom out
- **CTRL+0** Reset zoom

### 5.5.6 Experiment

- **CTRL+R** Recover experiment
- **CTRL+SHIFT+R** Open shared experiment
- **CTRL+SHIFT+R** Share experiment
- **CTRL+ALT+S** Save Manifest
- **CTRL+SHIFT+S** Save node login information (to CSV)

## D2.2: Federation operations, tools and support

- `CTRL+T` Terminate experiment
- `CTRL+ALT+T` Reboot experiment
- `F5` Update status
- `F6` Renew experiment
- `F10` Edit SSH-keys

### 5.5.7 Topology viewer

- `CTRL+L` Auto Layout
- `CTRL++` Zoom in
- `CTRL+-` Zoom out
- `CTRL+0` Reset zoom

### 5.5.8 Timeline viewer

- `ALT+R` Start
- `ALT+P` Pause
- `ALT+T` Stop
- `CTRL+I` Instant command
- `ALT+S` Save results
- `ALT+INSERT` Add command
- `CTRL+B` Add barrier

## 5.6 CAVEATS FOR SPECIFIC TESTBEDS

### 5.6.1 C-Lab

C-Lab uses a closed control network only accessible through VPN, and with IPv6 addresses. C-Lab nodes can be found under the wireless button in jFed.

In jFed you should do three things to enable SSH login while right clicking on a node:

- use jFed compile 1331 or later: <http://jfed.iminds.be/releases/r1331/> (click Experimenter GUI - Launch Webstart)
- use jFed proxy for SSH connections (see preferences)
- as C-Lab uses raw IPv6 addresses, if you use Putty on windows, you have to upgrade plink to the newest version: substitute C:\Program Files



## D2.2: Federation operations, tools and support

(x86)\PuTTY\plink.exe  
[snapshots/x86/plink.exe](http://tartarus.org/~simon/putty-snapshots/x86/plink.exe)

by [http://tartarus.org/~simon/putty-](http://tartarus.org/~simon/putty-snapshots/x86/plink.exe)

## 5.7 FREQUENTLY ASKED QUESTIONS (FAQ)

### 5.7.1 OS support

- jFed experimenter GUI now requires java 8 on all platforms. Only the oracle implementation is tested.
- Older versions will run on java 7, but only with at least Oracle Java 7 install (Update 55 or higher). On Linux, your mileage may vary with these older version, they tend to freeze on some Ubuntu's e.g.
- jFed currently does not support Open JDK. It should in theory work if you correctly install OpenJFX, but this is untested.

### 5.7.2 Install problems

Be sure to run a recent Java 8, verify with <http://java.com/verify> and/or `java -version`. If you have 2 java installations on your PC, you may revert to the command line start up of the Java 8 one, by downloading from <http://ifed.iminds.be/releases/> the latest release and take the `ExperimenterGUI - Download jar`.

Run this with something comparable to:

```
C:\Users\bvermeul\Downloads>"c:\Program Files (x86)\Java\jdk1.8.0_X\bin\java.exe" -  
jar "jFed-experimenter-GUI.jar"
```

This will help also to show you console output to detect what is wrong.

### 5.7.3 SSH login problems

- If you do not get connection to the node, try to enable the proxy support in the preferences
- If it still keeps saying `Server refuses key`, check in Putty on Windows that there is no fixed private key in the default settings of Putty

## D2.2: Federation operations, tools and support

(Connection - SSH - Auth - Private key file for authentication)

- If you used “Edit SSH Keys” but can’t login afterwards, one possible reason is that “Edit SSH Keys” needs to assign specific uid’s for the added users. So if any boot script or manual editing on the node has added a user without specifying a uid, the needed uid will not be available and sharing SSH keys won’t work.

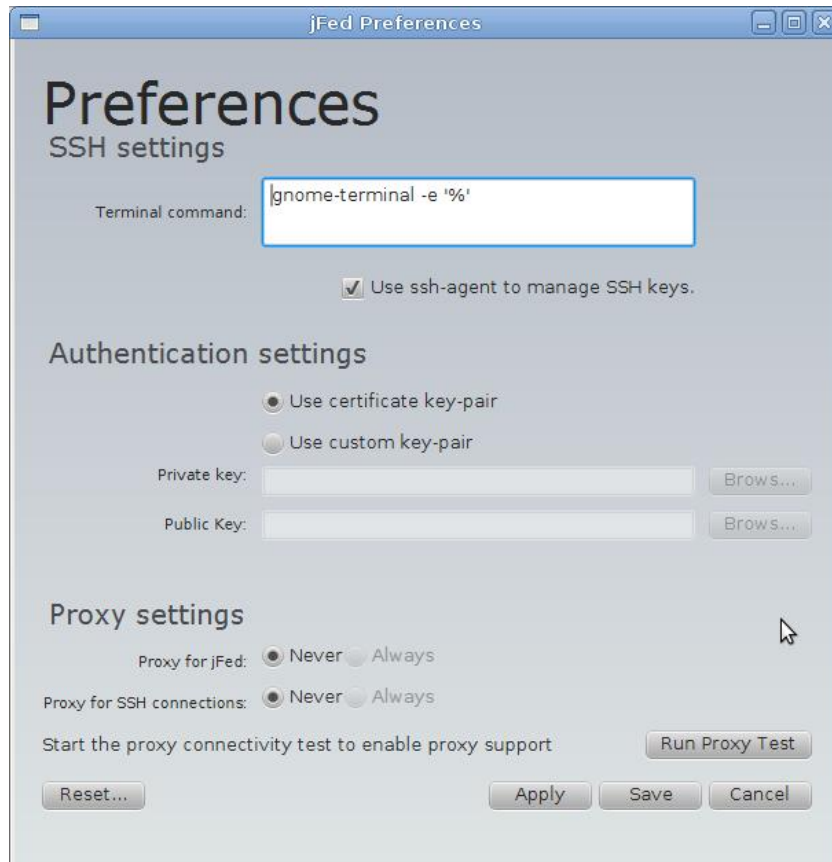
### 5.7.4Linux SSH terminal

Not all linux systems have the same SSH terminal, but if the default one does not work (when you right click a node when it is green, no SSH terminal opens), you can change terminals in jFed preferences as shown below.

Terminals to try:

```
konsole -e %  
  
gnome-terminal -e '%'      (quotes should be right !)  
  
xterm -e %  
  
x-terminal-emulator -e %
```

## D2.2: Federation operations, tools and support



The screenshot shows the 'jFed Preferences' dialog box with the following sections:

- SSH settings**
  - Terminal command:
  - Use ssh-agent to manage SSH keys.
- Authentication settings**
  - Use certificate key-pair
  - Use custom key-pair
  - Private key:
  - Public Key:
- Proxy settings**
  - Proxy for jFed:  Never  Always
  - Proxy for SSH connections:  Never  Always
  - Start the proxy connectivity test to enable proxy support

At the bottom, there are buttons for , , , and .



## 6 OTHER JFED SOFTWARE

jFed features a number of other tools next to the jFed experimenter GUI.

### 6.1 OVERVIEW

#### 6.1.1 GUI tools

The jFed Connectivity Tester is part of the jFed Experiment GUI, but can be used without logging in to the actual GUI. So you do not need an account to use it. The Connectivity Tester is useful to debug connectivity problems. It runs a number of connectivity tests, and then shows a report. This report can also be saved to disk.

The jFed Probe GUI is a tool for manually calling servers. Very low level calls can be made, and low level replies are visible. This tool requires knowledge of the communication APIs, and the tool itself is not a very user friendly GUI. It can be very valuable for testbed developers that wish to debug their server.

The Automated Testing GUI: This tool is used to run tests scenarios on servers. This is only useful for server developers, and it is not a very user friendly GUI. It includes simple tests, such as verifying if the `GetVersion` reply of an AM server is correct, and more complex tests, such as verifying if an AM can correctly provision a node, and if that node can be logged in to using SSH.

The jFed Scanner GUI: The jFed configuration contains information on each server that jFed can communicate with. This tool is used to scan a server and automatically fill in this info. It is of interest to server developers, and users that wish to use jFed with a server not known to jFed.

The Bugreport Viewer GUI: Is a standalone viewer for the bugreports that the jFed Experimenter GUI sends. This is only of interest to the jFed developers.

#### 6.1.2 CLI tools

The jFed Experiment CLI can execute some tasks from the command line, for full info, see [Experimenter CLI 1 \(legacy\)](#). There's also a second version, which exactly matches the actions jFed performs, and has some extra features, see [Experimenter CLI 2](#).

## D2.2: Federation operations, tools and support

The jFed Probe CLI has the same functionality as the jFed probe GUI, but it is a CLI instead of a GUI. This tool is not actively maintained, and not user friendly at all, even for a CLI. It can be useful for certain automation tasks. Contact the jFed authors if you think this tool is useful for you, or consider using [Geni's OMNI tool](#), or the jFed Experimenter CLI. The only advantage of the probe CLI over these tools is that it can send some m low level calls that the other tools cannot. An other alternative to using this CLI, is integrating the jFed library itself in you java project.

The Automated Testing CLI: This is the CLI version of the the Automated Testing GUI

The jFed Scanner CLI: A CLI tool related to the scanner GUI. This is no longer maintained.

### 6.1.3 Software libraries

The jFed core library is a java library that is the is the core of all jFed tools. It is open source and can be used for server communication.

The jFed RSpec library is a java library for RSpec parsing and manipulation. It is open source and can be used for RSpec manipulation.

The jFed Test Assistant is a library that allows java software to use jFed easily in unit or integration tests.

### 6.1.4 Fedmon

The Fedmon software suite powers <https://flsmonitor.fed4fire.eu>. This site monitors the status of servers in the Fed4Fire and Geni federation. This software is a seperate project that uses the automated testing library of the jFed software.

The last version of jFed also depend on the [Fedmon Web API](#) which is part of Fedmon.

## 6.2 TESTING CONNECTIVITY

### 6.2.1 What is the Connectivity Tester ?

You can follow the following procedure to test connectivity from a location (e.g. meeting room, hotel, ...). You don't need an account for testing this.

## D2.2: Federation operations, tools and support

### 6.2.2 Installing the Oracle Java JRE and start jFed

Follow the steps in the [Get Started](#) to install the Oracle Java JRE and start the green button 'Quickstart Experimenter GUI'. You don't need to install an SSH client for this test.

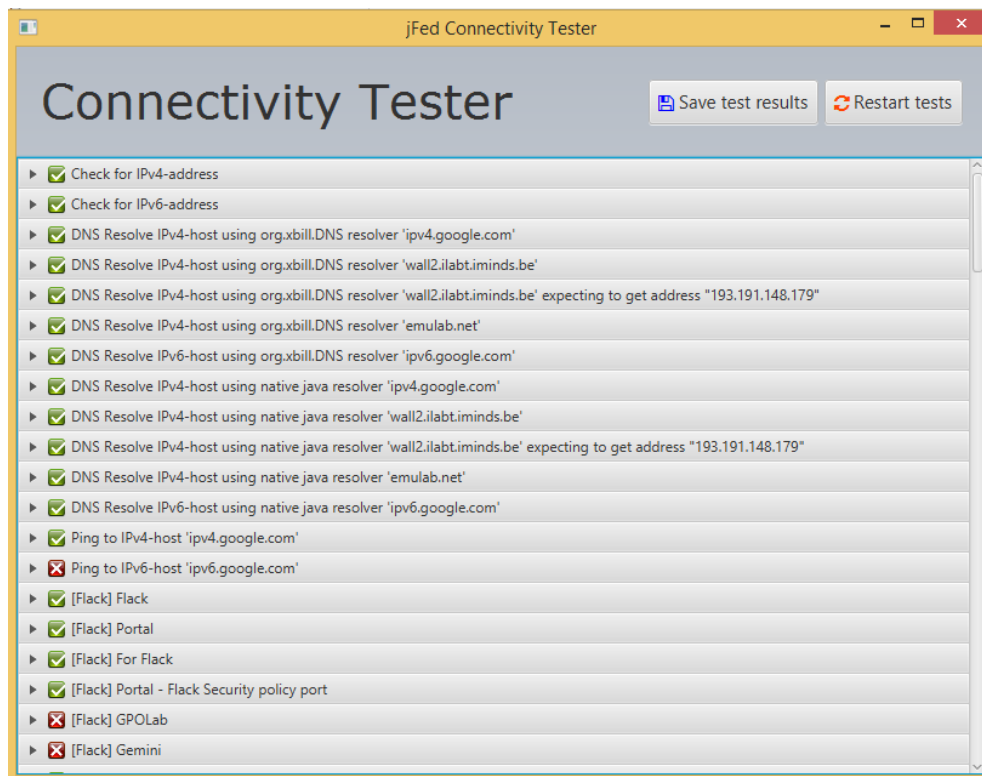
### 6.2.3 Starting the connectivity test

Then click the *Connectivity Tester* button at the bottom left, see screenshot below.



After the test has finished, you can click the 'Save test results' button, and save the results in a local file. The file can then be emailed e.g.

## D2.2: Federation operations, tools and support



## 6.3 EXPERIMENTER CLI 1 (LEGACY)

Note that this version of the experimenter CLI is no longer maintained, and might not be included in newer jFed releases. To avoid any confusion, the newer CLI will always be called `experimenter cli 2`. For more info on the new version of the CLI, see [Experimenter CLI 2](#).

### 6.3.1 Overview

The jFed experimenter CLI is used to create slivers easily from the command line.

The tool takes a command as first argument, and several options depending on the command.

The supported commands are:

`create`

Create one or more slivers. And optionally the slice used. Ansible playbooks can also be executed once everything is ready.

`delete`

Delete the specified slivers.

`status`

## D2.2: Federation operations, tools and support

Get the status of the specified slice.

`manifest`

Locally store the manifest RSpec of the specified slice (use `--manifest` to override the default location). This can also be used to generate [ansible](#) config files, using the `-ansible-dir` option, and to execute ansible playbooks.

`renew`

Renew the expiration time of the slivers

`userinfo`

Fetch user info: urn, slices and projects This command requires minimum version `develop 172` of the jFed CLI

`poa`

PerformOperationalAction. On emulab based testbeds this can be used to reload or restart nodes, and to get a console URL.

`createslice`

Create a slice, but allocate no resources (= create no slivers)

`slice-info`

Retrieve slice info. This also tries to retrieve a list of all slivers. It does not query the sliver status. (use `status` for that).

The first 4 commands are based around the user's request RSpec. So, `delete`, `status` and `renew` all need the request rspec as parameter. These commands also require slice information (to create or use a slice).

The `userinfo` command does not require as much information, as it only fetches user information.

The `poa` command requires a target sliver URN and an action to execute.

The `createslice` and `slice-info` command only require user and slice information.

### 6.3.2 Configuration

The CLI shares its configuration with the experimenter GUI. The easiest way to edit the configuration is using the experimenter GUI. You can also edit the file `~/.jFed/experimenter-ssh.properties` directly.

Typical settings to change are exogeni settings, and proxy settings.



## D2.2: Federation operations, tools and support

### 6.3.3 Download and Run

Go to [the jFed download page](#) and download the jFed CLI package at the bottom of the page. You can also [download a development version](#). Note that the top build on that page is the latest version. Click on the build of your choice, and on the next page, download the “jFed CLI (archive)”.

In both cases, you will download a file called `jfed_cli.tar.gz`. Extract this file to a directory and go to that directory. Here you will see `experimenter-cli.jar` and a `lib` dir. These are both required to run the Experimenter CLI.

Linux CLI example:

```
user@laptop ~/Downloads $ tar xzf jfed_cli.tar.gz
user@laptop ~/Downloads $ cd jfed_cli
user@laptop ~/Downloads/jfed_cli $ ls
automated-testing-5.2.0-SNAPSHOT.jar  experimenter-cli-5.2.0-SNAPSHOT.jar  lib  probe-cli-5.2.0-SNAPSHOT.jar
user@laptop ~/Downloads/jfed_cli $ java -jar experimenter-cli.jar
Syntax: jfed-experimenter-cli <command> [command_options ... ]
Available Commands: create,delete,status,renew
```

### 6.3.4 Usage

The tool is a java jar, so you need to use java to run it.

The create command takes a number of mandatory arguments:

```
-p <login PEM file>
```

The PEM login file that identifies you as an SFA user.

This file contains one or more certificates, and a matching private key.

Example content:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,B9AFE3F639D6D355

oDj5CQZ68EqY... (multiple lines) ...
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIEBzF... (multiple lines) ...
-----END CERTIFICATE-----
```

```
-P <PASSWORD>
```

## D2.2: Federation operations, tools and support

The plaintext password that locks the private key. If this option is not provided, and a key is requested, the password will be requested on `stdin` by the CLI.

Note that not all private keys are password protected and that you can remove the password from a private key using the `openssl` tool:

```
openssl rsa < password_protected_file.pem > unprotected_file.pem
```

(important: the `unprotected_file.pem` file will only contain the unprotected RSA key. Any certificate in `password_protected_file.pem` is not copied to `unprotected_file.pem`. This can be done manually)

The following options are mandatory for all but the `userinfo` command:

```
--rspec <rspec file>
```

The request RSpec that specifies the sliver(s) to be created.

```
--slice <slice name or URN>
```

The name of the slice in which the slivers(s) need to be created. This can be an existing slice, or a slice that needs to be created. In the later case, you need to also provide the `--create-slice` option in order to confirm that the experimenter CLI must create the slice for you. You may choose to either provide just the slice name, or the full slice URN. Both will work.

```
--project-name <project_name> or -S <project_name>
```

In case you used the `--slice` option to specify the slice name instead of the URN, you might want to specify the project name (sometimes referred to as *sub authority*). If you do not specify a project name, no project will be provided. Some authorities do not allow this. You can also specify the special value `CHOOSE_AUTOMATICALLY` as project name. In that case, the CLI will contact the SA and request a list of projects you are a member of, and automatically select the last project.

There are also a number of optional arguments. These are for settings for which the experimenter CLI has reasonable defaults. The arguments include:

```
--expiration-hours <integer>
```

## D2.2: Federation operations, tools and support

This option is used to specify when the slivers should expire. If a slice is created, this is the slice expiration time as well. This argument is specified as the number of hours from the current date. The default for this option is 2 hours.

Note that for both slice and sliver expiration time, the server might not honour your request, and return a longer or shorter time than what you requested.

### `--ssh-keys`

This option can be used to specify which ssh public keys should be added to the nodes when creating slivers. This option takes a comma separated list of public key source identifiers. The default value is: `usercert, rspec, shareduserallkeys` The 4 supported identifiers are:

#### `usercert`

The certificate used to authenticate contains a public key, which will be used as an ssh public key and added to created slivers. This way, you can login to the nodes using the matching private key.

#### `userkeys`

The server contains information on user public SSH keys. If this identifier is added, this information is requested, and the public ssh keys of the user are added when creating slivers.

#### `shareduserallkeys`

Add all ssh keys that are stored on the server for all users the slice is shared with. This will also request these users login certificate SSH key from the server.

#### `rspec`

The RSpec file can contain a list of public SSH keys. If this identifier is added, these are added when creating slivers.

To add public keys to an RSpec, add this as last child element of the `<rspec>` element:

```
<jfed-ssh-keys:user-ssh-keys user="urn:publicid:IDN+example.com+user+foo">
  <jfed-ssh-keys:sshkey>ssh-rsa AAAAB... foo@laptop</jfed-ssh-keys:sshkey>
```



## D2.2: Federation operations, tools and support

```
</jfed-ssh-keys:user-ssh-keys>
```

You may also need to add the correct namespace to the `<rspec>` element:

```
xmlns:jfed-ssh-keys="http://jfed.iminds.be/rspec/ext/jfed-ssh-keys/1"
```

Full example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rspec type="request"
  xsi:schemaLocation="http://www.geni.net/resources/rspec/3
http://www.geni.net/resources/rspec/3/request.xsd"
  xmlns="http://www.geni.net/resources/rspec/3"
  xmlns:jFed="http://jfed.iminds.be/rspec/ext/jfed/1"
  xmlns:jfed-ssh-keys="http://jfed.iminds.be/rspec/ext/jfed-ssh-keys/1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <node      client_id="node0"      component_manager_id="example.com+authority+cm"
exclusive="true">
    <sliver_type name="raw-pc"/>
  </node>

  <jfed-ssh-keys:user-ssh-keys user="urn:publicid:IDN+example.com+user+foo">
    <jfed-ssh-keys:sshkey>ssh-rsa AAAAB... foo@laptop</jfed-ssh-keys:sshkey>
  </jfed-ssh-keys:user-ssh-keys>
</rspec>
```

`--share-slice`

This options allows sharing the created or existing slice with other users. It expects a list of users, separated by commas. Users can be specified by either the user URN, or by the short username. You can also specify the special value "PROJECTUSERS", which will automatically share the slice with all users in the project. This is nice to use together with the `shareduserallkeys` option of `--ssh-keys` (which is set by default).

`--manifest <file>`

The file in which to store the manifest RSpec received when creating the sliver(s). This manifest typically contains details such as hostnames and SSH login details.

By default, the manifest is saved in `manifest-<SLICENAME>.rspec`

`--nowait`

By default, the experimenter CLI will wait until the slivers are ready before stopping. If this option is specified, it will not wait, but exit once the slivers have started initialising. (The manifest is saved in both cases).

## D2.2: Federation operations, tools and support

`--call-log <filename>`

This will store detailed information about all calls to a specified file. It is an XML based format.

`--slice-recover-info <filename>`

A file into which slice recover info will be stored. This can be used by the jFed GUI (if copied to the correct directory). (This functionality is not needed anymore if combining the latest CLI and GUI versions)

`--stitching`

If the RSpec requires stitching, allow it. By default, RSpecs that require stitching cause an abort

`--rewrite-rspec`

Parse the provided RSpec, and reconstruct it again, before sending it to the server. This can help prevent some server sides bug caused by valid but atypical RSpecs.

`--bind-rspec <component manager URN>`

Set the `component_manager_urn` for all “unbound” nodes in the provided request RSpec to the specified component manager. This option is allows using RSpecs that are not bound to a specific AM.

`--output-format`

The output format for the user info. Options are `text` and `json`. The default is `json`.

`--action`

and

`-target-sliver`

The options for the `poa` command.

`--delete-on-create-failure`

If there is a failure in a call made to create the sliver(s), delete all resources everywhere before exiting.

`--delete-on-become-ready-failure`

If there is a failure in a call while waiting for the sliver(s) to become ready, delete all resources everywhere before exiting.

An example:

```
java -jar experimenter-cli.jar create -s slice1 -S myproject --create-slice -p mypem.pem -P mypass --rspec lnode.rspec --expiration-hours 1
```

## D2.2: Federation operations, tools and support

Here, the user identified by *mypem.pem* create the slice *slice1* in project *myproject*, with a slice expiration of 1 hour (the server might use a longer slice expiration than requested). Then, a sliver is made, containing the node specified in *1node.rspec*. The CLI will wait until this sliver is ready, and will save the manifest RSpec in the file *manifest-slice1.rspec*.

To delete the created sliver you would use the command:

```
java -jar experimenter-cli.jar create -s slice1 -S myproject -p mypem.pem -P mypass --rspec 1node.rspec
```

To fetch user info (from the user in *mypem.pem*), you would use this command:

```
java -jar experimenter-cli.jar userinfo -p mypem.pem -P mypass
```

To reload a node, you would use this command:

```
java -jar experimenter-cli.jar poa --action restart --target-sliver urn:publicid:IDN+wall2.ilabt.iminds.be+sliver+73775 -p mypem.pem -P mypass -s slice1 -S myproject
```

To create a slice (and get text info instead of JSON info), you would use this command::

```
java -jar experimenter-cli.jar createslice --output-format text -p mypem.pem -P mypass -s slice1 -S myproject
```

To request slice info (in JSON format), you would use this command::

```
java -jar experimenter-cli.jar slice-info -p mypem.pem -P mypass -s slice1 -S myproject
```

### 6.3.5 Ansible support

There are a number of options which interact with [ansible](#).

The main option is used to write the ansible config files for the experiment:

```
--ansible-dir <dir>
```

This option is available for the `create` and `manifest` command. It writes ansible config files to the specified directory. The dir will be created if it doesn't exist. If it does exist, any existing files will be overwritten. The ansible files can be used with ansible

## D2.2: Federation operations, tools and support

commands, or with ansible playbooks. These can be used to create scripts that execute commands on different nodes. **Important:** In most cases (but not all), your private key will be copied to the created ansible dir. You should take the security considerations of this into account.

Next to ansible config files, [Fabric](#) files and ssh config files are also created. The ssh config file enables easy access to the nodes, even when an SSH proxy needs to be used. SCP access is also a lot easier using this file. An example:

```
~ $ cd ansible-files/  
~/ansible-files/$ ansible all -m ping  
~/ansible-files/$ ssh -F ssh-config node0  
~/ansible-files/$ scp -F ssh-config /work/file.tgz node0:/tmp/
```

A number of options is available to automatically execute ansible playbooks with the CLI:

```
--ansible-add-playbook <source [output_file]>  Add an ansible playbook to execute. The argument  
specifies the source, and optionally (after a space) the target (this option can be specified  
multiple times)  
--ansible-allow-any-playbook-outputfile          Allow any local file as execute_ansible_playbook  
output_file. (default: only use the basename of the specified output_file, and use the --ansible-  
dir as directory)  
--ansible-allow-playbook-inputfile              Allow local files in execute_ansible_playbook  
source. (default: only allow URLs as source)  
--ansible-debug                                Set debug flags when calling ansible playbook.  
--ansible-execute-rspec-playbooks              Execute all ansible playbooks found within  
request RSpec execute_ansible_playbook elements. (default: ignore them)  
--ansible-playbook-exe <arg>                  Executable for the ansible playbook. Default is  
"ansible-playbook" and is usually fine. On some systems might need to be  
"/usr/bin/python2/usr/bin/ansible-playbook"
```

As can be seen, ansible playbooks can either be specified on the command line, using the `--ansible-add-playbook` option, or they can be specified inside the request RSpec file. In the last case, the `playbook{s}` in the RSpec are only executed if the `--ansible-execute-rspec-playbooks` options is specified. To specify playbooks in the RSpec, add the following directly under the `rspec` element (so not in a node or link):

```
<jFed:execute_ansible_playbook                    xmlns:jFed="http://jfed.iminds.be/rspec/ext/jfed/1"  
source="https://example.com/playbook.yml" output_file="output.txt"/>
```

The output file will be stored in the dir specified by `--ansible-dir`, unless the output file is a full path and the `--ansible-allow-any-playbook-outputfile` options is specified. The source must be an URL, unless `--ansible-allow-playbook-inputfile` is specified, in which case it is also allowed to be the path to a local file.

`--ansible-playbook-exe` can be used to change the ansible command. Note that it allows spaces, so it can also be used to pass additional arguments to the ansible executable.



## D2.2: Federation operations, tools and support

### 6.3.6 Speaksfor credential support

Speaksfor credentials enable tools to execute calls on behalf of users, without having access to the user's private key or password. Speaksfor credentials are typically generated by a webpage on the user's authority, and passed to the tool. (The jFed probe can also generate speaksfor credentials, but that is not a typical use case.) The tool then includes these credentials in all calls to the AMs. If the provided speaksfor credential is valid, the AMs will execute the requested call on behalf of the user, instead of on behalf of the tool (and thus use the access rights of the user, not those of the tool). In the AM logs, it is visible everywhere that the tool speaks for the user. Note that speaksfor credentials have a limited validity, so tools cannot speak on behalf of the user forever.

Some servers support "chained" speaksfor. In this scenario, multiple speaksfor credentials are provided. For example, if valid speaksfor credentials are provided to allow tool A to speak for tool B, and tool B to speak for user C. Then, the AM will allow tool A to speak for user C.

The following command line option is used for speaksfor support:

```
--speaks-for <credential filename>
```

Include a speaksFor certificate in all calls, enabling you to make calls on behalf of another user.

This option can be specified multiple times, and/or the provided file may contain multiple speaksfor credentials. In these cases, multiple speaksfor credential are sent to the AM. The experimenter CLI will check scenarios with multiple speaksfor credentials, if the list of credentials forms no valid chain, it will fail before making any AM calls.

### 6.3.7 Command Line Help

If you do not enter any argument, the general syntax and the list of available commands will be returned. If you add a command as argument, but nothing else, specific help for the command will be returned.

Help for the `create` command:

```
user@laptop ~/Downloads/jfed_cli $ java -jar experimenter-cli.jar create
Command line argument Syntax error: Missing required options: s, r
usage: jfed-experimenter-cli <command> [command_options ... ]
Available commands: create,delete,status,renew,userinfo,sliceinfo,createslice
create options:
  --abort-if-slivers-exist          Do not create any slivers if any sliver already
exists on any of the authorities.
```



## D2.2: Federation operations, tools and support

```

-am,--am-api <AM API VERSION>           The AM Api version to use ("2" and "3"
supported, default: choose automatically)
--ansible-dir <DIR>                       The dir in which to save ansible config files.
Will be created if it doesn't exist. Will overwrite any existing files.
--authorities-file <AUTHORITIES XML FILE> The xml file containing the list of known
authorities. Default: choose automatically
--bind-rspec <COMPONENT MANAGER URN>      Bind any unbound nodes in the RSpec to a
specified authority. This leaves bound nodes as they are. (this implem --rewrite-rspec)
-c,--context-file <CONTEXT PROPERTIES FILE> The properties file containing context details
(only the login info in the context file is used by this tool)
--call-log <FILE>                         A file into which all call details will be
stored
--clearinghouse                           Fetch certificates etc from geni clearinghouse
first
--create-slice                             Create the slice if it doesn't exist. (if it
exists, this option is ignored)
-d,--debug                                 extra debugging output
--delete-on-become-ready-failure           If there is a failure in a call while waiting
for the sliver(s) to become ready, delete all resources everywhere before exiting.
--delete-on-create-failure                If there is a failure in a call made to create
the sliver(s), delete all resources everywhere before exiting.
-e,--expiration-hours <INTEGER>          The number of hours after which this slice
expires. This is an optional argument. default: 2 hours in the future
--expiration-date <RFC3339 DATE>         The date and time at which this slice expires.
In RFC3339 format. This is an optional argument. default: 2 hours in the future
--fake                                    Do everything, expect actually making the
slice and sliver calls. Useful for debugging syntax. (will make calls relating to retrieving
user data)
--fake-sliver                             Do everything, expect actually making the
sliver calls. (will make calls relating to retrieving user data and creating or retrieving
slice)
-k,--ssh-keys <OPTION LIST>              Specify which ssh keys to add. The argument
is a (comma separated) list of options.
Available options: usercert,userkeys,rsec,shareduserallkeys
usercert: add the user making the calls, with the ssh key from the certificate in the login PEM
file.
userkeys: add the user making the calls, with the ssh keys that are stored on the user authority
MA server. (usercert and userkeys can be combined to add all keys)
shareduserallkeys: add the ssh keys that are
stored on the user authority MA server for the users the slice is shared with AND the ssh keys
from that user's login certificate.
rsec: add the users and keys specified in the RSpec itself.
This option is optional, default: usercert,rsec,shareduserallkeys
-l,--logging                               activate logback logging output
--manifest <FILE>                         The file in which the manifest must be stored.
(default: manifest-<SLICENAME>.rspec)
--nowait                                  Do not wait until sliver is ready (default:
wait until ready)
-P,--private-key-password <CLEARTEXT PASSWORD> The password of the private key. Only used if
private key is password protected. Default: interactively ask password when needed.
-p,--cert-and-key-file <PEM FILE>         The file containing the user certificate and
private key (both in PEM format)
--print-calls                             Print all calls to stdout
-q,--quiet                                 less output
-r,--rspec <RSPEC XML FILE>              The rspec file to use for creating a sliver
--rewrite-rspec                           Parse the provided RSpec, and reconstruct it
again, before sending it to the server. This can help prevent some server sides bug caused by
valid but atypical RSpecs.
-S,--project-name <PROJECT NAME>         The name of the project (= sub authority) of
the slice. This is an optional argument (however some authorities might require a project!).
You can also use "CHOOSE_AUTOMATICALLY" as project name, in that case, the last (determined by
your SA) project you are a member of will be used.
-s,--slice <SLICE URN OR NAME>          The URN or name of the slice to use. (auto
detected)
--share-slice <USERNAME(S)>              List of users to share slice with. Either use
the URN of each user, or the short username. You can also specify the special value
"PROJECTUSERS", which will automatically share the slice with all users in the project. Multiple
users can be specified by separating them with a comma.
--slice-recover-info <FILE>              A file into which slice recover info will be
stored. This can be used by the jFed GUI (if copied to the correct directory).
--speaks-for <CERTIFICATE FILENAME>      The speaksFor certificate (in a file), enabling
you make calls on behalf of another user.
--stitching                               If the RSpec requires stitching, allow it. By
default, RSpecs that require stitching cause an abort

```

## D2.2: Federation operations, tools and support

```
-v,--version                show version and exit
```

Template Context Properties File:

```
username = <username>
passwordFilename = <filename of file containing password>
pemKeyAndCertFilename = <filename of file containing user certificate and private key in PEM
format>
userAuthorityUrn = <URN of test user's authority>
```

Help for the `delete` command:

```
Command line argument Syntax error: Missing required option: s
usage: jfed-experimenter-cli <command> [command_options ... ]
Available commands: create,delete,status,renew,userinfo,sliceinfo,createslice
delete options:
  -am,--am-api <AM API VERSION>                The AM Api version to use ("2" and "3"
supported, default: choose automatically)
  --authorities-file <AUTHORITIES XML FILE>      The xml file containing the list of known
authorities. Default: choose automatically
  -c,--context-file <CONTEXT PROPERTIES FILE>    The properties file containing context details
(only the login info in the context file is used by this tool)
  --call-log <FILE>                             A file into which all call details will be
stored
  --clearinghouse                               Fetch certificates etc from geni clearinghouse
first
  -d,--debug                                    extra debugging output
  --fake                                         Do everything, expect actually making the
slice and sliver calls. Useful for debugging syntax. (will make calls relating to retrieving
user data)
  --fake-sliver                                 Do everything, expect actually making the
sliver calls. (will make calls relating to retrieving user data and creating or retrieving
slice)
  -l,--logging                                  activate logback logging output
  -p,--cert-and-key-file <PEM FILE>             The file containing the user certificate and
private key (both in PEM format)
  -P,--private-key-password <CLEARTEXT PASSWORD> The password of the private key. Only used if
private key is password protected. Default: interactively ask password when needed.
  --print-calls                                 Print all calls to stdout
  -q,--quiet                                    less output
  -r,--rspec <RSPEC XML FILE>                  The rspec file to get the needed information
from (not mandatory for delete, but might not work without)
  -s,--slice <SLICE URN OR NAME>               The URN or name of the slice to use. (auto
detected)
  -S,--project-name <PROJECT NAME>             The name of the project (= sub authority) of
the slice. This is an optional argument (however some authorities might require a project!).
You can also
                                                use "CHOOSE_AUTOMATICALLY" as project name,
in that case, the last (determined by your SA) project you are a member of will be used.
  --speaks-for <CERTIFICATE FILENAME>          The speaksFor certificate (in a file), enabling
you make calls on behalf of another user.
  -v,--version                                  show version and exit

Template Context Properties File:
username = <username>
passwordFilename = <filename of file containing password>
pemKeyAndCertFilename = <filename of file containing user certificate and private key in PEM
format>
userAuthorityUrn = <URN of test user's authority>
```

### 6.3.8 Example speaks-for usage

You can use jFed CLI in speaks-as or speaks-for mode. Speaks-as is the typical way of using it where the user feeds its pem file (public signed certificate and private key) to jFed CLI and jFed CLI speaks 'as' that user. Things become more difficult if you want to run a service which uses jFed CLI to provision resources on a testbed. Of course, you can create a specific account

## D2.2: Federation operations, tools and support

and use that one for the service. So, all users using the service will have their resources created under the service account. As such, from testbed viewpoint, the service account is responsible for what happens with the nodes, and if quota are considered, the service account needs enough quota.

A second possibility is that users upload their private key to the service and the service uses that key to provision resources. This is not really nice from a security viewpoint.

The third possibility is that the service speaks-for the user. In that way, the service has its own certificate and private key, but the users give permission to the service to speak in their name (so the user trusts the service) without giving their private key ! They can also limit in time the duration for which the service can speak in their name, so to prevent abuse e.g.

### 6.3.9 Automatic link sharing

Emulab based testbeds allow users to share physical links between experiments/slice. Normally, a link has to be shared once the experiment is up. The jFed CLI offers a convenient way to set up this link sharing automaticall when creating an experiment. The following element has to be added to the `link` to be shared in the RSpec:

```
<jfed:auto_share_lan name="exampleSharedLan"/>
```

If the CLI `create` command is used to create an experiment containing a link with this element, the CLI will automatically share the link once all nodes and links are ready.

## 6.4 EXPERIMENTER CLI 2

Version 2 of the experimenter CLI replaces the version 1. To avoid any confusion, version 2 of the CLI will always be called `experimenter cli 2`. Version 2 is not completely backward compatible, but most command line options of version 1 are recognized, which should make converting not too hard.

### 6.4.1 Overview

The jFed experimenter CLI version 2 is used to create experiments easily from the command line, and to request some useful info about users and slices.

The tool has a mandatory `--action <action|file>` argument. The argument can either specify a file (which contains the action!), or specify an action.



## D2.2: Federation operations, tools and support

Extra configuration of the action can be specified in the file (if the action is specified using a file), or using command line arguments. If both are present, the command line arguments will overwrite the configuration in the file.

The supported actions are:

`run`

Create and run an experiment. This can use a number of RSpec/ESpec sources, and creates a new slice. Ansible playbooks can be executed after the experiment is ready (ESpecs can also execute ansible playbooks, in separate earlier step).

`delete`

Delete the specified slivers.

`status`

Get the status of the specified slice.

`manifest`

Locally store the manifest RSpec of the specified slice (use `--manifest` to override the default location). This can also be used to generate [ansible](#) config files, using the `-ansible-dir` option, and to execute ansible playbooks.

`renew`

Renew the expiration time of the slivers

`userinfo`

Fetch user info: urn, slices and projects This command requires minimum version `develop 172` of the jFed CLI

`poa` or `performOperationalAction`

PerformOperationalAction. On emulab based testbeds this can be used to reload or restart nodes, and to get a console URL.

`createslice`

Create a slice, but allocate no resources (= create no slivers)

`sliceinfo`

Retrieve slice info. This also tries to retrieve a list of all slivers. It does not query the sliver status. (use `status` for that).

`sliceCredential`

To be implemented: fetch the slice manifest and store it

`decrypt`

To be implemented: decrypt the user login PEM file and store it

## D2.2: Federation operations, tools and support

Each of these command requires different info. You can get help, including an example for each command easily:

```

user@laptop ~/Downloads $ java -jar experimenter-cli2.jar --action poa --help
usage: experimenter-cli --action sliceinfo
  -c,--context-file <FILE>          Specify the "context" file. This old format contains info
  about the actual login file(s).It is advised to use the -p option instead of this one.
  -C,--cert-file <FILE>            Specify the file containing the "login" X509 certificate, in
  PEM format
  --call-log <FILE>                A file into which all call details will be stored
  -d,--debug                        extra debugging output
  -h,--help                          show help and exit
  -k,--key-file <arg>              alias for --cert-and-key-file    (Only present for backward
  compatibility. Might be removed in future versions)
  --manifest <FILE>                The file in which the manifest must be stored. (default:
  manifest-<SLICENAME>.rspec)
  --output-format <FORMAT>         The output format to use (for user information). Choices:
  "text" or "json". Default:json
  -p,--cert-and-key-file <FILE>    The file containing the user certificate and private key
  (both in PEM format)
  -P,--privkey-pass <PASSWORD>    The password of the private key. Only used if private key is
  password protected. Default: interactively ask password when needed.
  -pa,--print-action                Take no action, but instead output the action file matching
  the current instructions (which are made up out of cli argument and/or the content of the action
  file)
  --print-calls                     Print all calls to stdout
  -q,--quiet                         less output
  -s,--slice <SLICE URN/NAME>       The URN or name of the slice to use. (auto detected)
  -S,--project-name <NAME>         The name of the project (= sub authority) of the slice. This
  is an optional argument (however some authorities might require a project!). You can also use
  "CHOOSE AUTOMATICALLY" as project name, in that case, the last (determined by your SA) project
  you are a member of will be used.
  --silent                           less output (same as --quiet)
  -v,--version                       show version and exit

Example yml for "sliceinfo":
---
action: SLICEINFO
showMetaInfo: true
showStatus: true
showUsers: true
slice:
  expireTimeMin: 120
  failIfNoProject: true
  failOnExistingSlice: false
  project: my_project_s_name
  projectSource: PROVIDED
  sliceName: my_slice_name
user:
  password: my_pem_password
  passwordMethod: DIRECT
  pem:
  - user.pem
  speaksForCredential: []
  
```

## 6.4.2 Configuration

The CLI shares its configuration with the experimenter GUI. The easiest way to edit the configuration is using the experimenter GUI. You can also edit the file `~/ .jFed/experimenter-ssh.properties` directly.

Typical settings to change are exogeni settings, and proxy settings.

## D2.2: Federation operations, tools and support

### 6.4.3 Download and Run

Go to [the jFed download page](#) and download the jFed CLI package at the bottom of the page. You can also [download a development version](#). Note that the top build on that page is the latest version. Click on the build of your choice, and on the next page, download the “jFed CLI (archive)”.

In both cases, you will download a file called `jfed_cli.tar.gz`. Extract this file to a directory and go to that directory. Here you will see `experimenter-cli2.jar` and a `lib` dir. These are both required to run the Experimenter CLI.

Linux CLI example:

```

user@laptop ~/Downloads $ tar xzf jfed_cli.tar.gz
user@laptop ~/Downloads $ cd jfed_cli
user@laptop ~/Downloads/jfed_cli $ ls
automated-testing.jar  experimenter-cli.jar  probe-cli.jar  experimenter-cli2.jar  lib
user@laptop ~/Downloads/jfed_cli $ java -jar experimenter-cli2.jar
Syntax: jfed-experimenter-cli [-a|--action <ACTION NAME | ACTION FILE>] [action_options ... ]
Available Actions:
run,createSlice,performOperationalAction,renew,delete,userinfo,sliceinfo,sliceCredential,manif
est,decrypt
Help for a specific command: jfed-experimenter-cli --action <ACTION NAME> --help
  
```

### 6.4.4 Usage

The tool is a java jar, so you need to use java to run it.

The advised method to use the tool, is to create an “action file” (in yaml format) and pass that to the action argument of the tool. This provides the full range of options the CLI has to offer. Alternatively, you can provide some of the options using command line arguments (for the most part the same as for version 1 of the cli).

The easiest way to get info on both the “action yml” format, and the command line options, is to request help for the specific action, for example:

```

user@laptop ~/Downloads/jfed_cli $ java -jar experimenter-cli2.jar --action sliceinfo --help
usage: experimenter-cli --action sliceinfo
  -c,--context-file <FILE>          Specify the "context" file. This old format contains info about
the actual login file(s).It is advised to use the -p option instead of this one.
  -C,--cert-file <FILE>            Specify the file containing the "login" X509 certificate, in
PEM format
  --call-log <FILE>                A file into which all call details will be stored
  -d,--debug                       extra debugging output
  -h,--help                         show help and exit
  -k,--key-file <arg>              alias for --cert-and-key-file    (Only present for backward
compatibility. Might be removed in future versions)
  --manifest <FILE>                The file in which the manifest must be stored. (default:
manifest-<SLICENAME>.rspec)
  --output-format <FORMAT>        The output format to use (for user information). Choices:
"text" or "json". Default:json
  
```

## D2.2: Federation operations, tools and support

```

-p,--cert-and-key-file <FILE>   The file containing the user certificate and private key (both
in PEM format)
-P,--privkey-pass <PASSWORD>   The password of the private key. Only used if private key is
password protected. Default: interactively ask password when needed.
-pa,--print-action              Take no action, but instead output the action file matching
the current instructions (which are made up out of cli argument and/or the content of the action
file)
  --print-calls                 Print all calls to stdout
-q,--quiet                      less output
-s,--slice <SLICE URN/NAME>     The URN or name of the slice to use. (auto detected)
-S,--project-name <NAME>       The name of the project (= sub authority) of the slice. This
is an optional argument (however some authorities might require a project!). You can also use
"CHOOSE_AUTOMATICALLY" as project name, in that
case, the last (determined by your SA) project you are a member
of will be used.
  --silent                      less output (same as --quiet)
-v,--version                    show version and exit

```

Example yml for "sliceinfo":

```

---
action: SLICEINFO
showMetaInfo: true
showStatus: true
showUsers: true
slice:
  expireTimeMin: 120
  failIfNoProject: true
  failOnExistingSlice: false
  project: my_project_s_name
  projectSource: PROVIDED
  sliceName: my_slice_name
user:
  password: my_pem_password
  passwordMethod: DIRECT
  pem:
  - user.pem
  speaksForCredential: []

```

The recommended usage is to create an action file, and pass only that file and the user info on the command line. For example, create this file called `run_experiment.yml`:

```

action: RUN
experiment:
  requestRSpec:
    source: PROVIDE_CONTENT
    providedContentSource: |
      <rspec xmlns="http://www.geni.net/resources/rspec/3" type="request">
        <node client_id="node0" exclusive="false"
component_manager_id="urn:publicid:IDN+docker.ilabt.imec.be+authority+am">
          <sliver_type name="docker-container"/>
        </node>
      </rspec>
  slice:
    sliceName: expl
    failOnExistingSlice: false
    expireTimeMin: 120
    projectSource: PROVIDED
    project: myProject
  waitForReady:
    maxTimeMin: 5
shareWith:
  projectMembers: true
deleteOn:
  failCreate: true
  failBecomeReady: true
  failConnectivityTest: true

```

And run it with this command:

## D2.2: Federation operations, tools and support

```
user@laptop ~/Downloads/jfed_cli $ java -jar experimenter-cli2.jar --action run_experiment.yml  
-p login.pem
```

This will create a slice names “exp1” in the project “myProject” with a single docker AM node, which expires in 2 hours. The slice will be shared with all members of the project “myProject”. The public SSH keys of these members will be added to the node. If the resources are not ready in 5 minutes or fails in another way, the slice will be deleted.

