

The Business Of Filing Apache Derby Issues In Jira

Note from the author: As the current volunteer administrator of the Derby project in Jira, I intend this document to help those not very familiar with Jira and/or Derby when filing/updating issues. Feedback on the document will be greatly appreciated.

- Ron Reuben (rreuben@sourcery.org)

Introduction

This document is suggested reading for anyone who wants to enter an issue against Derby. At Apache, the following issue types have been defined in Jira (Derby's issue tracking system):

Bug – A problem impairing functionality, performance, reliability or documentation (a defect)

New Feature – A request for a new feature that has yet to be developed

Improvement – An improvement/enhancement of an existing feature

Test – An entry regarding a test (unit, integration, system, stress etc.) including requests for new tests

Wish – A generic “wishlist” item (e.g. INFRA-58 which was a wish to have consistent date formats for the “Created” & “Updated” dates displayed at the top of an issue filed in Jira)

Task – A generic task that does not fit into any of the categories above (not very common)

This document explains the following:

- How to search Jira to identify if the issue you are reporting has already been filed
- What information helps complete the issue entry
- Understand some of the non-obvious fields in Jira's “new issue” entry screen
- The difference between “Resolve” and “Close”
- How to provide useful bug reproduction instructions and/or scripts

Once you have read through this document or are familiar with its contents, you may access Jira at <http://issues.apache.org/jira>

New Issue Entry

Before filing a new issue, it is recommended that you first search Jira to verify that it has not yet been reported.

How to search Jira

Once logged in to Jira, click on the “Find Issues” link as indicated in the image below:

ASF JIRA

In the screen that follows, click the “New” tab at the top (unless you have saved a query in the past; explained below). Select “Derby” as your project. You may want to fill in the “Fix For”, “Matches Versions” & “In Components” fields to narrow your search. You will most likely want to enter a search keyword in the “Text Search” field labeled ‘Query’. In the screenshot below, we are looking for the keyword “Mac” in the 4 searchable fields – Summary, Description, Comments & Environment. There are other options that you may specify to narrow the search but these are optional. Click “View” to have Jira perform your search.

Filter: [View](#) [New](#) [Manage](#)

You do not currently have a search or filter selected.

<< View & Hide

View >>

Project: 

Selection automatically reloads page

Project Components

(only available if there is a single project selected)

Fix For:
No Fix Version
Unreleased Versions

In Components:
No Component
Build tools

Matches Versions:
No Version
Unreleased Versions

Text Search

Query: 

Query Fields: Summary Description
 Comments Environment

Issue Attributes

Issue Type:
Bug
New Feature

Reported By:

: 

Assignee:

Issue Na

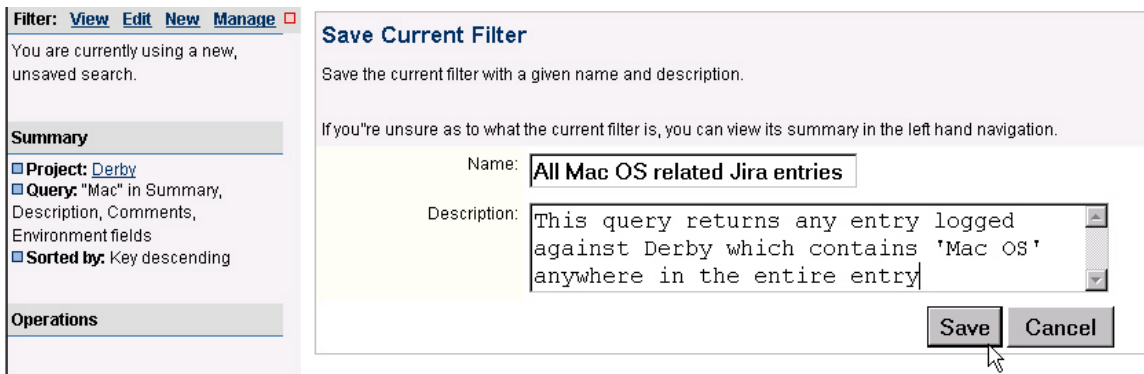
The Issue N

Using the p

If the query you just performed is new and likely to be useful in the future, you may want to save it as a filter. To do this, click the “Save” link as shown in the screenshot below.



Enter a name for your filter and a relevant description. For our example query filter, we enter the details as specified below.



Once you click “Save”, the filter is saved as a Private filter that only you can see. You can manage this filter (edit, delete etc) by clicking on the “Manage” link as seen below



Note that there are many options available to search Jira; please choose those most relevant to your issue at hand. Choose keywords carefully to increase the chance you will find the issue if it has already been reported.

If your search reveals the issue has already been filed, please update the entry with any relevant information you may be able to add. See the section below, titled “What makes a Derby Jira entry really useful,” for a list of items that help create a ‘useful-to-the-developer’ issue entry.

If the issue has not been reported, you may want to post your issue on the derby-user@db.apache.org mail list, asking the following questions:

- If you are reporting a bug, is your bug really a bug (or a restriction, feature request etc)
- Has anyone else noticed this issue and if so, under what circumstances
- Can anyone suggest a workaround for the issue
- If relevant, why you think the bug is a “blocker” (highest priority that needs to be fixed right away)

This is optional, but recommended and should not be seen as a replacement to filing the issue in Jira.

Filing a new issue

To file a new issue, click the “Create New Issue” link on the Jira home page. This presents you with the screen below. You now select the project (‘Derby’ in this case) and issue type. For a description of the various types of issues supported by Jira, see the section titled ‘Introduction’ at the top.

Create Issue

Step 1 of 2: Choose the project and issue type...

* Project: Derby

* Issue Type: Bug

Next >>


This site is running on JIRA in a free Open Source backing application. Eval

Click “Next”. The subsequent screen that appears will be used to enter you basic issue information. Let’s analyze portions of the screen. The screenshot below assumes you chose the issue to be a “Bug” in the previous screen. Enter a short, to-the-point summary of the bug issue being reported. Try to include key words into your summary, making it easy to find when a search is issued. Choose the priority of the issue.


Create Issue

Step 2 of 2: Enter the details of the issue...

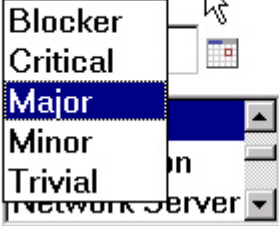
Project: Derby

Issue Type:  Bug

* Summary:

Priority: **Major** 

Due Date:

Component/s: **Major** 

The Jira system at Apache defines 5 priority levels:

- Blocker: Show-stopper! Blocks development and/or test work. No work-around possible. Needs to be worked on right away.
- Critical: Severe issue that potentially has a work-around but may not be acceptable under all circumstances. Includes issues such as system crashes, loss of data or severe memory leaks. New features & improvements in this category are required urgently but not asap.
- Major: Issue has a work-around but there is a major loss of functionality. For new feature/improvements, this needs to be implemented soon (but not urgently)
- Minor: Some loss of functionality. Work-around exists. For new feature/improvements, there is no urgency with the implementation of this request.
- Trivial: Lowest priority. No loss of functionality. No hurry to fix/code this. Mostly a cosmetic issue like misspelled words or misaligned text

Next, select the component and versions the issue is being filed against. The “Affects Version/s” is the version/s that was being used when it was detected. Derby defines the following components:

Unknown – to be used when you do not know the right component (remember any "developer" can update your entry later)

Build tools – for issues with the build scripts

Documentation – for issues with Derby documentation

JDBC – for issues that can be attributed to the JDBC driver (including XA)

Localization – when localization is the issue

Network Server – for network server related issues (including servlet)

Security – for issues related to security


Services – for issues with Derby code related to basic services (e.g. cache, lock manager etc.)

SQL – for issues that can be attributed to query parsing/processing/optimizing etc

Store – for issues related to the storage of data on disk (i.e. disk storage module)

Test – for issues with the tests

Tools – for issues with ij, dblook, import/export & sysinfo

Due Date: 

Component/s: **JDBC**
Localization
Network Server

Affects Version/s: **Unknown**
Unreleased Versions
- 10.0.2.0

Fix Version/s: **Unknown**
Unreleased Versions
- 10.0.2.0

The “Assign To” field shows who should own the fix to the issue being reported. You will only assign someone to the issue if you have had prior agreement with the person to fix it or you are assigning the issue to yourself.

Assign To: **- Automatic -**

Environment: **- Automatic -**

Description: **A B**
Daniel John Debrunner
Jalud Abdulmenan
Jean T. Anderson
Kathey Marsden
Mamta A. Satoor
Myrna van Lunteren
Ramandeep Kaur
Ron Reuben

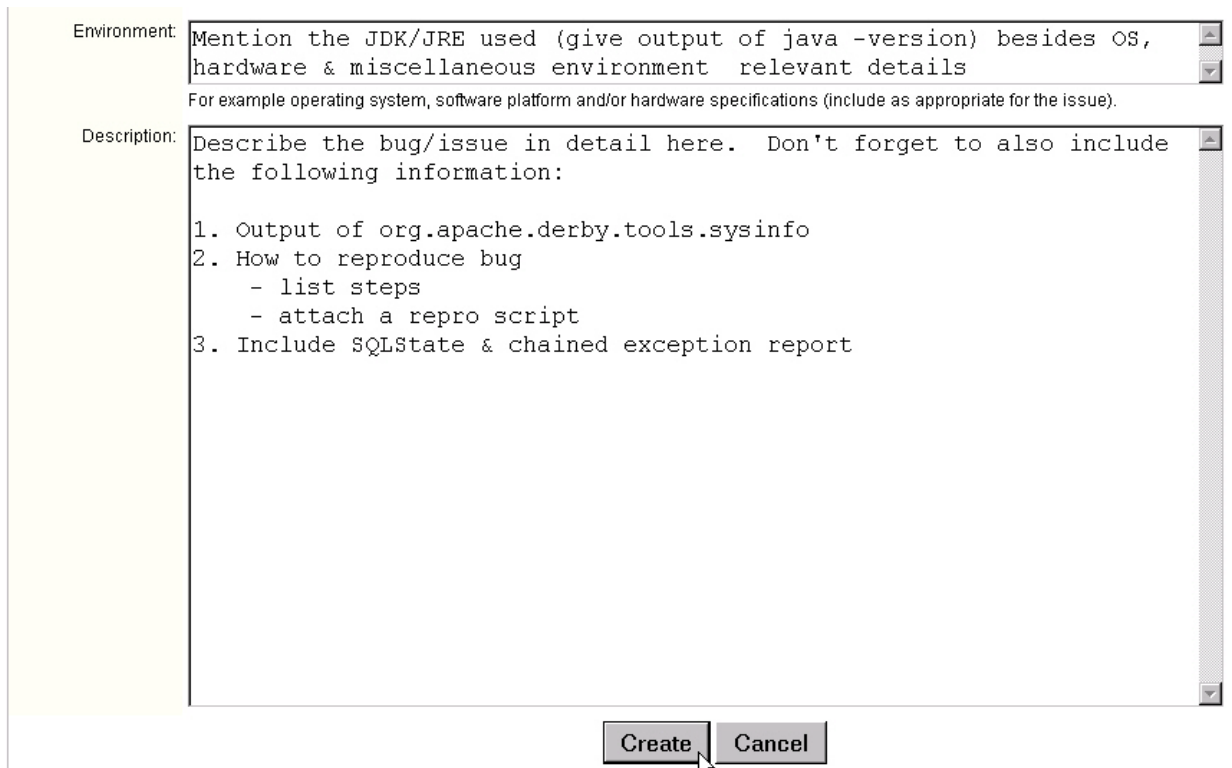
re platfc

Next, make entries in the “Environment” and “Description” fields. In the former, you should give all relevant details that will help the assigned developer fix the issue. This includes information such as:

- Hardware platform
- Software environment
- JDK/JRE being used (give the output of `java -version` if appropriate)

The “Description” field should be used to give all possible details of the issue in question. If the issue you are filing is a bug, please include the following details in this field:

- Output of `org.apache.derby.tools.sysinfo`
- How to reproduce the bug (either step-by-step information or attach a reproduction script/program)
- Chained nested exceptions reported by Derby and the SQLSTATE reported by the system



The screenshot shows a form with two main text input areas. The first is labeled 'Environment:' and contains the text: 'Mention the JDK/JRE used (give output of java -version) besides OS, hardware & miscellaneous environment relevant details'. Below this text is a smaller line of text: 'For example operating system, software platform and/or hardware specifications (include as appropriate for the issue)'. The second area is labeled 'Description:' and contains the text: 'Describe the bug/issue in detail here. Don't forget to also include the following information:'. Below this is a numbered list: '1. Output of org.apache.derby.tools.sysinfo', '2. How to reproduce bug' (with sub-points '- list steps' and '- attach a repro script'), and '3. Include SQLState & chained exception report'. At the bottom of the form are two buttons: 'Create' and 'Cancel'.

Once you are done, click the “Create” button to log your issue in Jira.

To attach a file, program or screenshot to the issue you filed, you will click the “Attach file” or “Attach screenshot” links on the screen that describes your issue.

| Operations | Description |
|---|---|
| <input type="checkbox"/> Assign this issue (to me) <input type="checkbox"/> Attach file to this issue <input type="checkbox"/> Attach screenshot to this issue <input type="checkbox"/> Comment on this issue | Consider a simple case of - A table tbl has 10000 rows, there is a primary key and the query in question is <pre>select * from tbl where i1 in (-1,100000)</pre> derby does a table scan of the entire table even if |

How to edit an issue

Once you file an issue, you can edit it only if you have “developer” access to the Derby project. To gain “developer” access, send a mail to the derby-dev@db.apache.org email list. To edit an issue, first open the issue in Jira, then look for the “Operations” section on the left side of the screen. Here, you will find options to “Edit”, “Link”, “Comment” etc. Please do not delete an issue unless you are absolutely sure of what you are doing. To comment on an issue, click “Comment”, to edit any information contained in the issue, click the “Edit” link, to assign the issue to yourself or some other developer (assuming you have agreement with the developer to do this), you click the “Assign” link. The “Link this issue” option is meant to annotate the fact that the issue in question is either a duplicate of or has a dependency relationship with another issue already filed in Jira.

| Operations | |
|---|--------|
| <input type="checkbox"/> Assign this issue (to me) | s h |
| <input type="checkbox"/> Attach file to this issue | |
| <input type="checkbox"/> Attach screenshot to this issue | |
| <input type="checkbox"/> Comment on this issue | T |
| <input type="checkbox"/> Delete this issue | |
| <input type="checkbox"/> Edit this issue | |
| <input type="checkbox"/> Link this issue to another issue | |
| <input type="checkbox"/> Move this issue to another project | |
| <input type="checkbox"/> Voting: You have not voted for this issue. Vote for it if you wish it to be fixed | |
| <input type="checkbox"/> Watching: You are not watching this issue. Watch it to be notified of changes | |

Resolve an issue

Once an issue has been addressed appropriately, you need to “Resolve” it. Do not close the issue! “Close” and “Resolve” are separate “workflow actions” identified. You will ideally first resolve an issue before the submitter closes it.

To resolve an issue click on the “Resolve Issue” link on the left side of the screen that appears once you open the issue.

| | |
|--|--|
| Available Workflow Actions | |
| <input type="checkbox"/> Resolve Issue | Environment: a |
| Operations | |
| <input type="checkbox"/> Assign this issue (to me) | Description Consider a simple case of - A table tbl has 10000 rows, there is a pri and the query in question is select * from tbl where i1 in (-1,100000) |
| <input type="checkbox"/> Attach file to this issue | |
| <input type="checkbox"/> Attach screenshot to this issue | |

In the resulting screen, you will update the resolution to one of the following values:

- Fixed – Issue has been fixed as per your understanding. The fix has been tested and checked in. Remember to add a comment with relevant details that explain your fix.
- Won't fix – This issue will not be fixed for reasons explained in the comments field
- Duplicate – Issue in question is a duplicate of an existing issue. You should “Link” this issue to the existing entry and make a mention of the issue id in the comments field
- Invalid – The issue is invalid for the project in question. The submitter needs to re-evaluate the entry and perhaps re-submit the issue with additional details (if relevant)
- Incomplete – Insufficient information to work on the issue
- Cannot Reproduce – Usually set by the developer who tries to fix this issue. Indicates that the issue cannot be reproduced. Comments are an absolute must to explain the steps you took

The “Fix Version” entry is relevant if the resolution of an issue is “Fixed” or “Duplicate”.

Resolve Issue

Resolving an issue indicates that the developers are satisfied the issue is finished.

If you have permission to close issues, you can optionally resolve & close the issue at the same time.

* Resolution:

Fix Version/s:

(Select the versions that this issue is actually fixed in).

Assign To:

Comment: (an optional comment describing this update)

Update comment:

Comment Viewable By:

Once you enter the required information, you will usually just “Resolve” the issue. If you are confident that the issue needs to be closed as well (see below), click the “Resolve and Close” (if the option is presented). One would usually only “Resolve and Close” if the fix has already been verified by the submitter or the person Resolving is the submitter.

Resolving an issue changes the “State” of the issue to “Resolved” from “Open” or “In Progress”.

Note: One can always re-open an issue once it is resolved. This marks the status to “Open”

Close an issue

A developer who fixes/codes an issue will usually only “resolve” it to one of the states mentioned above. It is the submitter’s responsibility to “close” the issue once the fix/code is verified. Closing an issue is the final workflow step in the life of an entry in Jira. To close an issue, click the “Close Issue” link of the issue, as seen below.

Issue Details

Key: [DERBY-30](#)

Type:  Bug

Status:  Resolved

Resolution: Fixed

Priority:  Minor

Assignee: [Daniel John Debrunner](#)

Reporter: [Daniel John Debrunner](#)

Votes: 0

Available Workflow Actions

[Close Issue](#)

[Reopen Issue](#)

As seen in the screen below, it will be nice to enter a comment before you click “Close”.

Close Issue

Closing an issue indicates there is no more work to be done on it, and it has been verified as complete.

Assign To:

Comment: (an optional comment describing this update)

Update comment:

Comment Viewable By:

Closing an issue sets the status of the issue to “Closed”. This indicates there is no more work to be done on it and that the issue has been verified as complete.

Note: One can always re-open an issue once it is closed. This sets the status back to “Open”

What makes a Derby Jira entry “really useful”

This mainly pertains to bug entries. The 2 key parts of a bug entry are the reproduction script/program and the bug description.

Reproduction scripts/programs

A developer who attempts to fix a bug will find it much easier if the entry contains instructions or scripts/programs that help reproduce the bug. The following kinds of reproduction information will prove useful:

- Scripts – Given that the community will be working on a diverse set of platforms, it will be best to provide a SQL script that reproduces the bug in question. For example, a Perl or Shell script may not be useful to someone working on the Windows platform without additional developer packages installed. When providing a SQL script, please try to insert relevant comments that assist with readability and understanding.
- Programs – Sometimes, a JDBC application program may help reproduce an issue. Attach the file if relevant
- Code snippets – include any relevant code snippets in the description or attach these to the bug entry
- Step-by-Step instructions – This approach is usually highly effective. Make sure your instructions are precise and include all the relevant steps you did when uncovering the bug.

Bug descriptions

As part of the bug description, you should also enter the following information if relevant:

- ‘sysinfo’ details – Run the ‘org.apache.derby.tools.sysinfo’ tool and include the output of the command with your description (see below for a sample screenshot). This contains important version, CLASSPATH, build, JRE and platform information that will be useful to a developer trying to reproduce and resolve the bug.
Please run the following command in the exact same environment your Derby application ran under when the bug was detected
`‘java org.apache.derby.tools.sysinfo’`
When you set your application environment manually, make sure you do the very same setup before running *sysinfo*. On the other hand, when the application sets up the environment, you need to run *sysinfo* under the application environment to ensure that accurate information is captured by *sysinfo*.
- Nested Exception information – Derby will most likely generate nested exceptions when an error is detected. When you catch an exception in a JDBC application, remember to iterate through the chained exceptions one by one and capture these in the output of your program (*getNextException()* method). When an exception is thrown on the console window, please copy-paste (or screen-capture) the entire output (including nested exceptions) into the bug entry
- SQLSTATE – Derby will generate a SQLSTATE along with an exception. Do not forget to capture this information in your bug entry.

- Include any relevant output from the derby.log file. If this information is too confusing for you to understand and the file isn't too big, you may want to attach it to your bug entry

```
$ java org.apache.derby.tools.sysinfo
----- Java Information -----
Java Version:      1.4.2_03
Java Vendor:      Sun Microsystems Inc.
Java home:        d:\p4clients\ronr-tp_main\jdk142\jre
Java classpath:   ./derby.jar;./db2jcc.jar;./db2jcc_license_c.jar
OS name:          Windows 2000
OS architecture: x86
OS version:       5.0
Java user name:   ronr
Java user home:   C:\Documents and Settings\ronr
Java user dir:    D:\ronr\pantry\Derby\v10.0.2
----- Derby Information -----
[D:\ronr\pantry\Derby\v10.0.2\derby.jar] 10.0.2.0 - (46005)
[D:\ronr\pantry\Derby\v10.0.2\db2jcc.jar] 2.4 - (17)
[D:\ronr\pantry\Derby\v10.0.2\db2jcc_license_c.jar] 2.4 - (17)
-----
----- Locale Information -----
-----
$ _
```