# POConfItemBuddy 1.0.0

## Owner's Manual

Revised September 22, 2012
Terrance A. Crow

## Introduction

Project Open's Conf Item/inventory engine is powerful on its own. With it, you can track all kinds of inventory items, from hardware servers to software to hardware and software components. Conf Items are smart enough to prompt only for the important fields based on the type of item, so that a Hardware Server might prompt for an IP address, whereas a Software Application would not. All in all, Project Open's Conf Items subsystem offers a robust set of features.

One piece of functionality that I didn't see -- and that I needed -- was managing Conf Item relationships. Granted, Project Open (]po[) can already link a Conf Item to a person, Help Desk ticket, or a project. However, I wanted to do more. I wanted the following features:

1. Ability to logically group servers, like all Windows servers or all Linux servers
2. Ability to track what software runs on what server, including application servers (like Tomcat or Enterprise Service Bus)
3. Ability to track what applications run under a given server's application server (like a Tomcat servlet or an Enterprise Service Bus JBI workflow or OSGi bundle)

To implement this, I considered writing web application, but I don't know enough about AOLServer to write a secure web application from scratch. I considered writing a package in ]po[, but I didn't have time to learn the framework. I passed on extending the existing ]po[ functionality because I don't know enough TCL. The latter approach would have been my preference so I could have contributed back to the project. Unfortunately, I just didn't have that kind of time. So, I settled on a different approach.

I previously wrote PONotificationAssistant, an application that sends e-mail to individuals who are assigned a given Task in a ]po[ workflow. That application uses Java. I took a similar approach here. The difference is that I wanted a decent UI for this application (PONotificationAssistant has no UI). To solve that issue, I used NetBeans 7.2's form engine to produce the Java UI elements.

Be warned: the code is profoundly ugly. Though I've coded in Java for years, I've never done a UI outside of a JSP or servlet-generated HTML page. So, please don't expect this to be pretty or elegant or anything except one thing: functional.

**An apology to the ]po[ development team:** Project Open might very well be able to do all of this, and I just don't know how to make it work. Plus, I fully admit that I should have taken the time to develop this within ]po[ so I could have contributed back to the project. So, please don't take this effort as any kind of insult! It's based on my own shortcomings, though I do hope it's useful.

## Overview of the Solution

I wanted this solution to run from my ]po[ server, which runs Linux. However, as explained earlier, I didn't want to write it as a web application. I want to run it from the server so I don't have to allow network-based access to the ]po[ server's PostgreSQL database, because I consider that an unwarranted security exposure. In other words, I want to keep the ]po[ server's configuration as close to the default as possible.

POConfItemBuddy is a Java application, so it requires the Java Virtual Machine. The installation instructions below include steps to install Java. If you already use PONotificationAssistant, you already have Java installed, so you won't need to install it again.

POConfItemBuddy interacts directly with PostgreSQL, so it requires the PostgreSQL JDBC Type 4 driver. The procedures later include steps to install the driver.

The default PostgreSQL security configuration won't allow even the local server to attach via IP, which is how the PostgreSQL JDBC driver attaches. You'll need to make a minor change to a key PostgreSQL security configuration that'll allow the local server to interface with the database. The installation procedures include steps to handle this.

POConfItemBuddy has a small properties file that you'll need to update to point to your ]po[ server. This document includes those procedures, too.

Before trying to run POConfItemBuddy, you'll need to create three new tables in your ]po[ database (projop). I took this approach to minimize the potential for colliding with how ]po[ is already working with the projop database. Both the source distribution and the binary distribution include the scripts you'll need to create those tables and their indexes. This document includes procedures to handle those tasks.

Finally, you'll need to install POConfItemBuddy's Java jar and its support library. You can either download the source package and compile it using your favorite IDE (it's packaged for Netbeans 7.2, within which you can just select Run -> Clean and Build) or download the pre-compiled binary.

You can find both at Source Force. Here's the URL:

`https://sourceforge.net/projects/poconfitembuddy/files/`

Since POConfItemBuddy is written in Java and uses Java UI classes, you'll need to connect to the Linux host using a graphics-enabled client. This chart summarizes how to do that based on what client you have:

| Client Type | Expand Product | Notes |
|---|---|---|
| Linux | Linux: just use ssh (ssh -l <your ID> -X <hostname> | A Linux workstation can connect to another Linux workstation/server and run graphical UIs from the host if you include the -X parameter (under most distributions) |
| OS X | Depends on version; for Mountain Lion, use XQuartz | Once you install the X Windows emulator, you can connect with the same syntax as Linux |
| Windows | MobaXTerm | MobaXTerm is a robust program that allows you to establish a non-graphical or graphical host connection using the same syntax as Linux. |

*Table 1: Ways to Connect to your ]po[ server to run POConfItemBuddy*

If you install POConfItemBuddy on a Windows server, you can connect to that Windows server using Remote Desktop Connection (RDC).

One it's all installed, here's now POConfItemBuddy supports relating Conf Items:

1. Group Servers: POConfItemBuddy lets you logically group servers any way you like. For example, you might to create a group that contains all of your Linux Server or a group that identifies the servers related to a certain business unit. Then, as you enter Help Desk tickets and identify the hardware as one of these groups, you can track actions against a whole group of servers without entering multiple tickets. You can also write reports that'll show what tickets affect what individual servers. Reports examples are later in this manual.
2. Relate software to servers: You can say what software runs on what server. For example, you could say that SQL Server runs on one server, that Project Open runs on another server, that some monitoring software runs on both, and that Tomcat runs

on a third. You reports could then show not only what servers were affected by an outage; you could see what software was affected.

3. Relate individual applications to software containers: Some software, like Tomcat or Microsoft's Internet Information Server (IIS), can themselves run other software programs. For example, your corporate development team might write a program that runs under Tomcat. You might purchase a package that runs under IIS. POConfItemBuddy lets you relate these individual products to another piece of software. Then, your reports could show not only what servers were affected by a ticket; the report could show individual software/software contains and any custom applications they contain.

You can find more details on the table configuration in Appendix A.

**Note:** POConfItemBuddy works by linking the Nr field on Conf Items. This means your Nr values should be unique to each record. If they're not, please do yourself a favor and make them unique before trying to use POConfItemBuddy.

# Installing POConfItemBuddy

These instructions assume you are starting with a Linux box -- specifically, the downloadable image that the ]po[ team makes available. If you're running PO on another platform, you should be able to make it work -- the precise steps will just not be the same as the ones presented here.

**Install Java**

Probably the fastest way to install Java is to open a terminal session on your ]po[ server, become root, and type:

```
yum install java
```

On Linux distributions like CentOS 5.x or Ubuntu, this should install the OpenJDK version of Java. POConfItemBuddy should work fine with that.

Optionally, you could install the official Oracle version of Java according to the instructions that Oracle provides.

**Install POConfItemBuddy**

If you want to compile POConfItemBuddy, do so now. I used Netbeans 7.2 to build it, and I've included the Netbeans project file directory in the source code version of the zip.

You can download either package from the Source Forge site:

https://sourceforge.net/projects/poconfitembuddy/files/

These instructions assume you're going to use the binary zip. On your ]po[ server, while logged in as root, create this directory:

```
/usr/local/POConfItemBuddy
```

Still as root, unzip the contents of the binary zip into this new directory. You should now have the following files in /usr/local/POConfItemBuddy:

1. `lib/`
   1.1. `lib/swing-layout-1.0.4.jar`
2. `POConfItemBuddy.jar`
3. `POConfItemBuddy.properties`
4. `README.TXT`
5. `Scripts`
   5.1. `Scripts/c1_conf_item_container_linkage.sql.sql`
   5.2. `Scripts/c1_conf_item_linkage.sql`
   5.3. `Scripts/c1_conf_item_meta_defs.sql.sql`

Edit this file:

```
POConfItemBuddy.properties
```

You should see this value:

```
jdbcurl=jdbc:postgresql://ohcmhsrv024.interstell.home:5432/
projop
```

Change "ohcmhsrv024.interstell.home" to "127.0.0.1".

If you'd like to see if the basics are right, try this command:

```
java -jar POConfItemBuddy.jar
```

You should see a screen that looks like this:

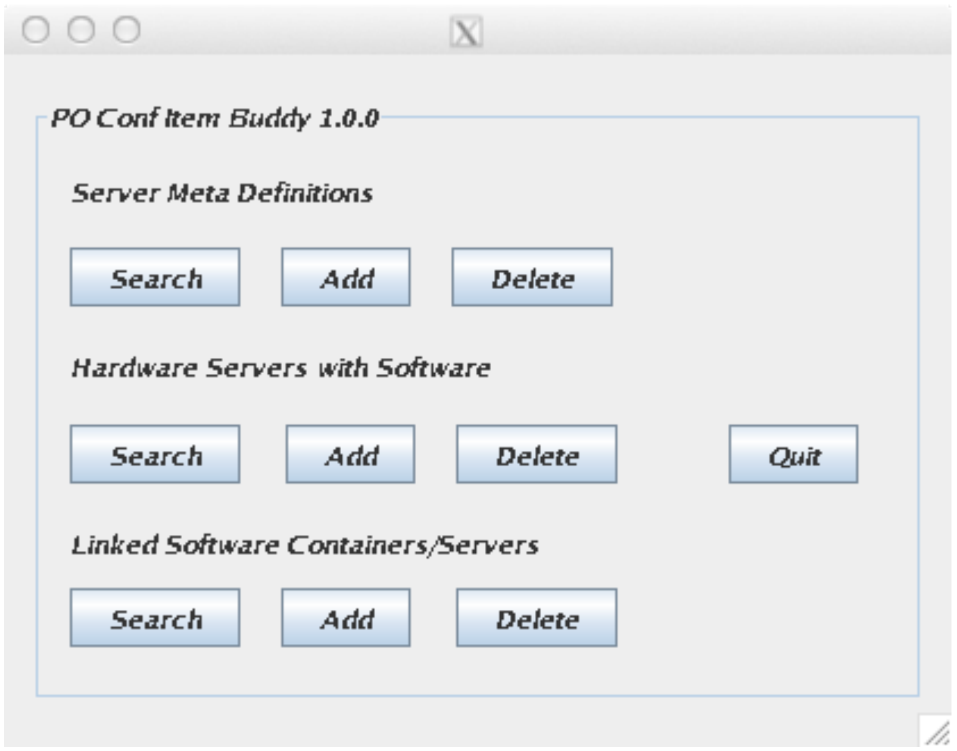


*Figure 1: The main POConfItemBuddy screen.*

If you see the screen from Figure 1, it means the basic framework is working. Click on this button:

```
Quit
```

It's time to install the database driver.

**Install PostgreSQL Driver**

Visit the PostgreSQL web site and download the latest JDBC4 driver. At the time of this writing, you could find it here:

```
http://jdbc.postgresql.org/download.html
```

We now have to place it where the Java runtime will find it. Since we're running the POConfItemBuddy.jar with the -jar option, we can't use the -cp command line switch. So, I cheat: I put the jar in the Java JRE lib/ext directory. For the OpenJDK under CentOS 5.x, that means I put the PostgreSQL JDBC4 driver here:

```
/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0/jre/lib/ext
```

## Load the New Tables

The POConfItemBuddy comes with the scripts you'll need to load the new tables into ]po[. First, become the projop user:

```
su -l projop
```

Change into the POConfItemBuddy directory where the scripts are contained:

```
cd /usr/local/POConfItemBuddy/Scripts
```

First, create the meta data table that'll hold the server groupings:

```
psql -f c1_conf_item_meta_defs.sql.sql
```

Create the table that'll link software to servers:

```
psql -f c1_conf_item_linkage.sql
```

Finally, create the table that'll link applications to software or software application servers:

```
psql -f c1_conf_item_container_linkage.sql.sql
```

Exit out of the projop user and back into the root user:

```
exit
```

While it's possible to run POConfItemBuddy as root, it's bad form. So, if you don't already have a local user ID on your ]po[ server, you should create one now. I'll create tcrow and set its password:

```
adduser tcrow
passwd tcrow
```

Most of you already know this, but the Linux user has nothing to do with the ]po[ user.

**Note:** If you're using Windows for your ]po[ server, you can create either a domain account or a local user account. If the former, make sure it has rights to log into the server, probably with Terminal Services or RDC.

At this point, you should be ready to fire up POConfItemBuddy! Before we start with a tour, let's make sure the platform's solid. As root, change into this directory:

```
/usr/local/POConfItemBuddy
```

Issue this command:

```
java -jar POConfItemBuddy.jar
```

Click on this button under “Server Meta Definitions:”

```
Search
```

Make sure you see a blank screen and not a pop-up error message. If you get an error that says POConfItemBuddy can't access your server, check this file:

```
/var/lib/pgsql/data/pg_hba.conf
```

Near the bottom, you should see a section like this:

```
# IPv4 local connections:
#host    all         all         127.0.0.1/32          ident sameuser
host    all         all         127.0.0.1/32       trust
```

In the ]po[ image I downloaded, the example shows what I saw. If yours is missing the third line and the second line is not commented out, it might be that permissions aren't set right for this to work. Make a copy of pg_hba.conf and change it to make it look like the example above. Then, restart PostgreSQL

```
service postgresql restart
```

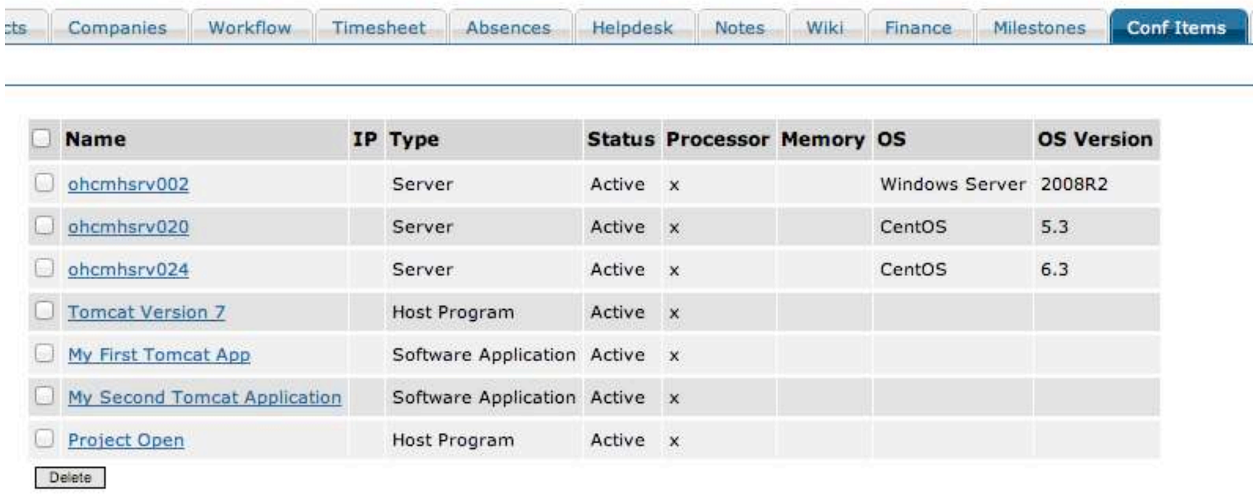
It's a good idea to restart ]po[ after restarting PostgreSQL:

```
killall -9 nsd
```

Wait a few minutes for ]po[ to come back on line. Now you should be ready for a tour!

## A Tour of POConfItemBuddy

Before taking a tour, let's set the stage. Let's say we have a simple ]po[ installation that includes hardware and software configuration information. Figure 2 shows an example of our Conf Items:



| | Name | IP | Type | Status | Processor | Memory | OS | OS Version |
|---|---|---|---|---|---|---|---|---|
| ☐ | ohcmhsrv002 | | Server | Active | x | | Windows Server | 2008R2 |
| ☐ | ohcmhsrv020 | | Server | Active | x | | CentOS | 5.3 |
| ☐ | ohcmhsrv024 | | Server | Active | x | | CentOS | 6.3 |
| ☐ | Tomcat Version 7 | | Host Program | Active | x | | | |
| ☐ | My First Tomcat App | | Software Application | Active | x | | | |
| ☐ | My Second Tomcat Application | | Software Application | Active | x | | | |
| ☐ | Project Open | | Host Program | Active | x | | | |

*Figure 2: Here's our simple Conf Item listing*

In this example, we have three servers:

1. ohcmhsrv002
2. ohcmhsrv020
3. ohcmhsrv024

We have two stand-alone applications:

1. Tomcat Version 7
2. Project Open

We have two web applications that run under Tomcat:

1. My First Tomcat App
2. My Second Tomcat applications

We'll walk through establishing the relationships between these Conf Items for our tour.

If you're still logged into your ]po[ server from the installation tasks above, log out. From a Linux terminal session on another server/workstation, from an OS X terminal session, or from a MobXTerm session, enter this command:

```
su -l tcrow -X ohcmhsrv020
```

For "tcrow", substitute your Linux ID on the ]po[ server. For "ohcmhsrv020", substitute your server's hostname.

Once you're logged in, change into the target directory:

```
cd /usr/local/POConfItemBuddy
```

Start the application:

```
java -jar POConfItemBuddy.jar
```

You should see the same screen that you did in Figure 1, above.

## Scenario 1: Setting Up a Meta Server

If you look carefully at our list of Conf Items in Figure 2, you'll see that one is a Windows server is two are Linux servers. Let's say that I want a way to say that a Help Desk ticket affects all of the Linux servers in our environment. Maybe I want to record a Linux patch cycle, or that I want to deploy a piece of monitoring software to all of our Linux servers. Further, let's say that I want to enter a single Help Desk ticket for the entire operation, and yet still record that I worked on all of the Linux servers. The first thing to do is setup a special type of Conf Item by creating a new hardware Conf Item Type.

Go to the Admin tab and click on Categories.

Click on Intranet Conf Item Type.

Add a category called:

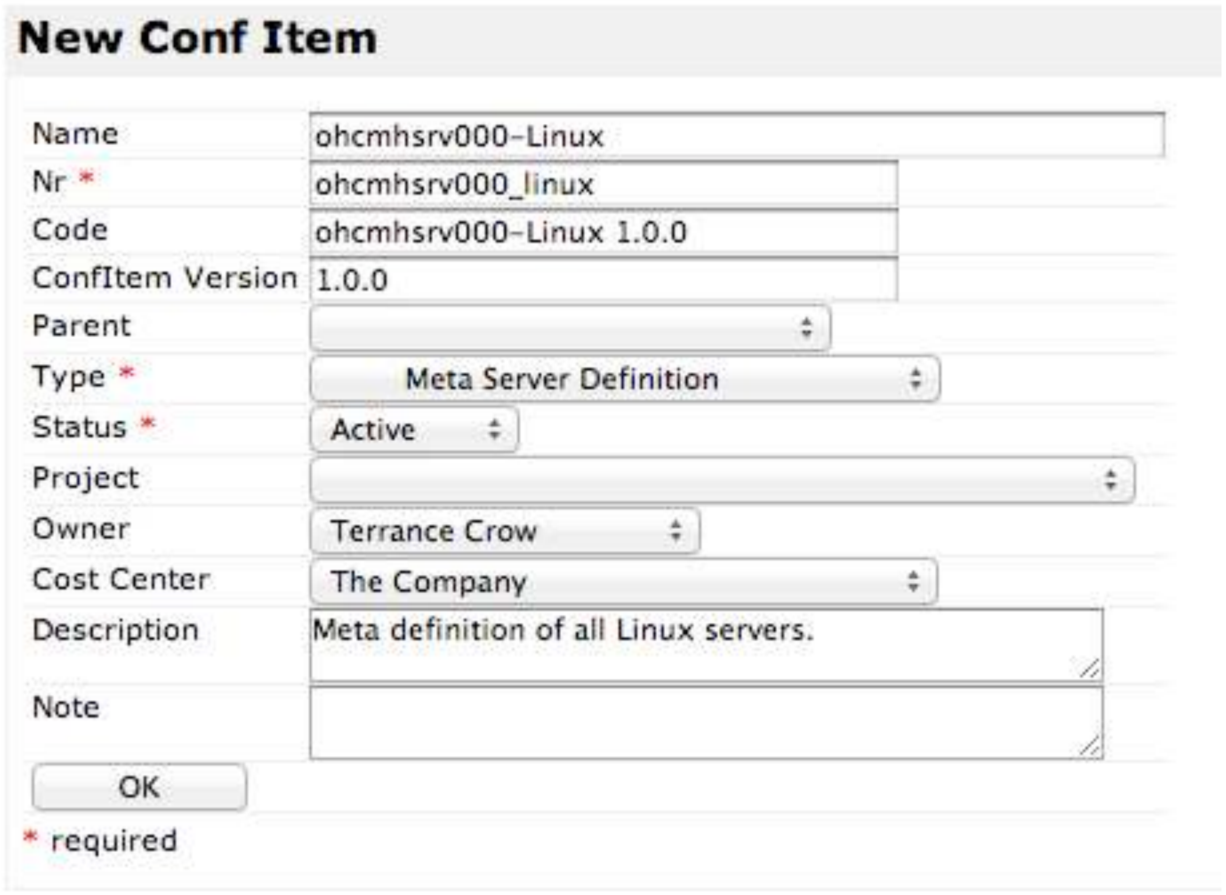
```
Meta Server Definition
```

Fill in all of the Category translations for the languages your version of ]po[ needs to support. Save the item, then click on its URL to edit it. For Parent Categories, click on:

```
Hardware
```

Save the changes.

The next step is to setup a Conf Item that'll be the meta server definition. I'll create a new Conf Item that looks like the one in Figure 3.



New Conf Item

| | |
|---|---|
| Name | ohcmhsrv000-Linux |
| Nr * | ohcmhsrv000_linux |
| Code | ohcmhsrv000-Linux 1.0.0 |
| ConfItem Version | 1.0.0 |
| Parent | |
| Type * | Meta Server Definition |
| Status * | Active |
| Project | |
| Owner | Terrance Crow |
| Cost Center | The Company |
| Description | Meta definition of all Linux servers. |
| Note | |

*Figure 3: Setting up the meta server definition.*

After you save it, it should be visible on your Conf Item list.

Before setting up the meta server, it's important to remember that POConfItemBuddy works on the Nr item. So, if we're going to relate the two Linux servers to ohcmhsrv000-Linux, we need to know these three Nr values:

| Conf Item | Nr |
|---|---|
| ohcmhsrv000-Linux | ohcmhsrv000_linux |
| ohcmhsrv020 | ohcmhsrv020 |
| ohcmhsrv024 | ohcmhsrv024 |

With those three bits of information, we can get to work. From the POConfItemBuddy main screen, under the heading "Server Meta Definitions," click on this button:

```
Add
```

You should see a screen that looks like Figure 4.

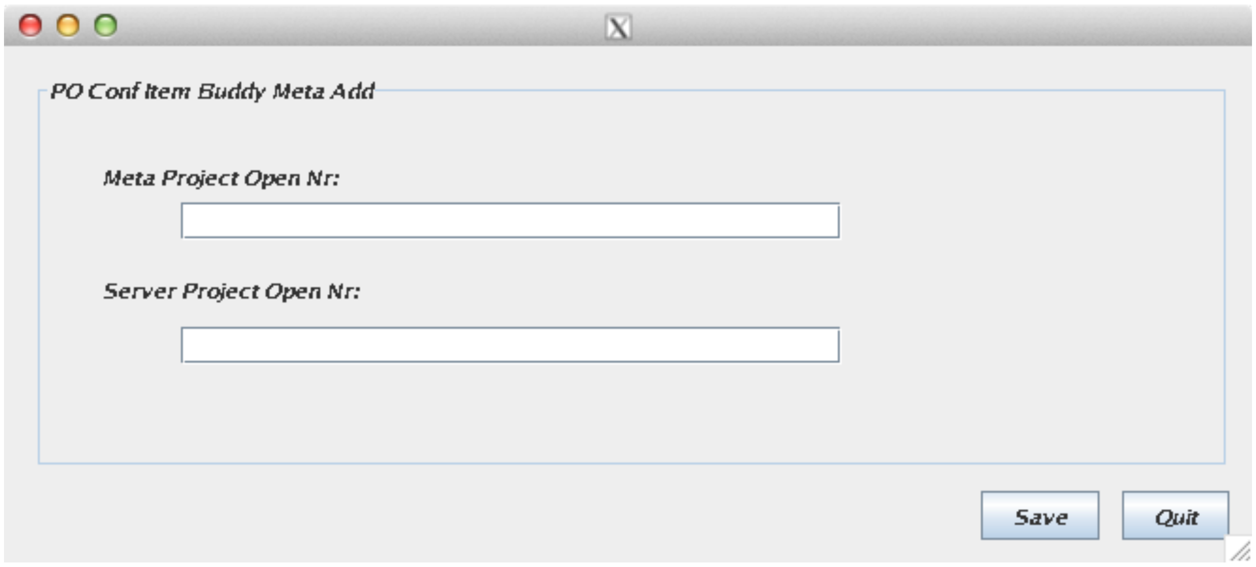


*Figure 4: POConfItemBuddy tries to keep things simple by prompting for the minimum amount of data necessary.*

To setup the first relationship, enter "ohcmhsrv000_Linux" as the "Meta Project Open Nr" and "ohcmhsrv020" as the "Server Project Open Nr." Click on this button:

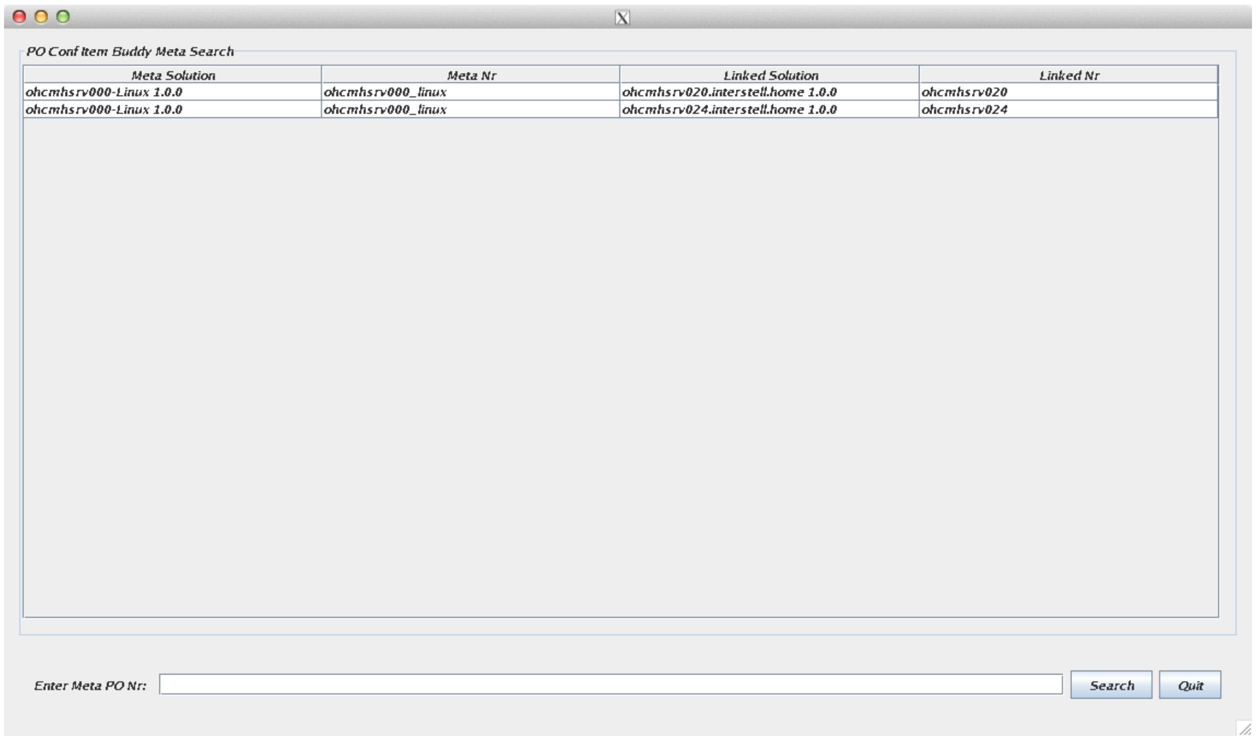
```
Save
```

Before it saves, POConfItemBuddy confirms that the Nr for both Conf Items exists. It'll also try to make sure you haven't already entered this relationship. POConfItemBuddy then tries to save the item. It'll share the results of its attempts with you via a dialogue box.

If you repeat the process for "ohcmhsrv000_linux" and "ohcmhsrv024", you should be able to check to see if the entries are there. From the main menu, under "Server Meta Definitions," click on this button:

```
Search
```

You should see a screen that looks like Figure 5.


PO Conf Item Buddy Meta Search

| Meta Solution | Meta Nr | Linked Solution | Linked Nr |
|---|---|---|---|
| ohcmhsrv000-Linux 1.0.0 | ohcmhsrv000_linux | ohcmhsrv020.interstell.home 1.0.0 | ohcmhsrv020 |
| ohcmhsrv000-Linux 1.0.0 | ohcmhsrv000_linux | ohcmhsrv024.interstell.home 1.0.0 | ohcmhsrv024 |

*Figure 5: POConfItemBuddy shows you an unfiltered list at first.*

If you have a small Conf Item inventory, you really won't need to perform any searches. On the other hand, if you have hundreds of servers, you can filter them on the meta Nr (in this case, ohcmhsrv000_linux) to show only the members of that server meta group.

Also, you can click on the heading of all four columns to sort by that column.

So far, so good. We now have a meta definition setup to identify our Linux servers. Let's see what else POConfItemBuddy can do.

**Scenario 2: Mapping Software to Servers**

In figure 2, you can see that we have two "standalone" pieces of software: Tomcat Version 7 and Project Open. Let's say for the sake of our example that Tomcat Version 7 runs on ohcmhsrv024 and Project Open runs on ohcmhsrv020. We already know the Nr values for the two servers (they're the same as the server names). That means we need to know the Nr values for Tomcat and ]po[. Looking at the individual Conf Items, we know the values are:

| Software | Nr |
|---|---|
| Tomcat Version 7 | tomcat_version_7 |

| Software | Nr |
|---|---|
| Project Open | project_open |

On the main POConfItemBuddy screen, under "Hardware Servers with Software," you should click on this button:

`Add`

You should see a screen that looks like the one in Figure 6:

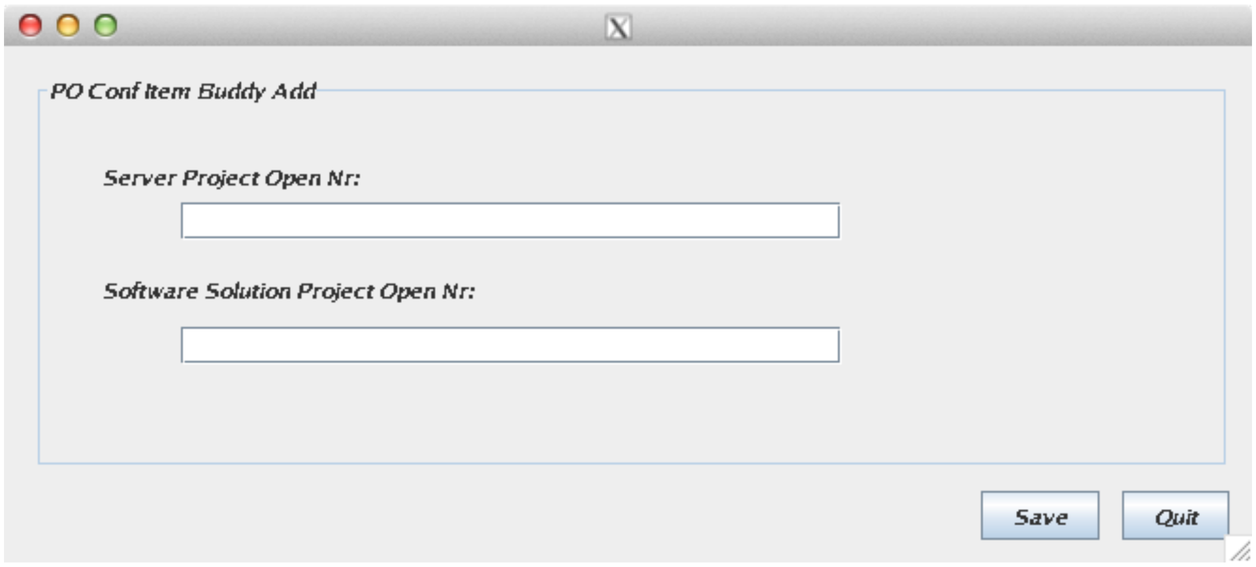


*Figure 6: The screen to relate software to a server looks remarkably the same as the screen to link servers to meta servers.*

In this screen, enter "ohcmhsrv024" as the Server Project Open Nr and "tomcat_version_7" as the Software Solution Project Open Nr.

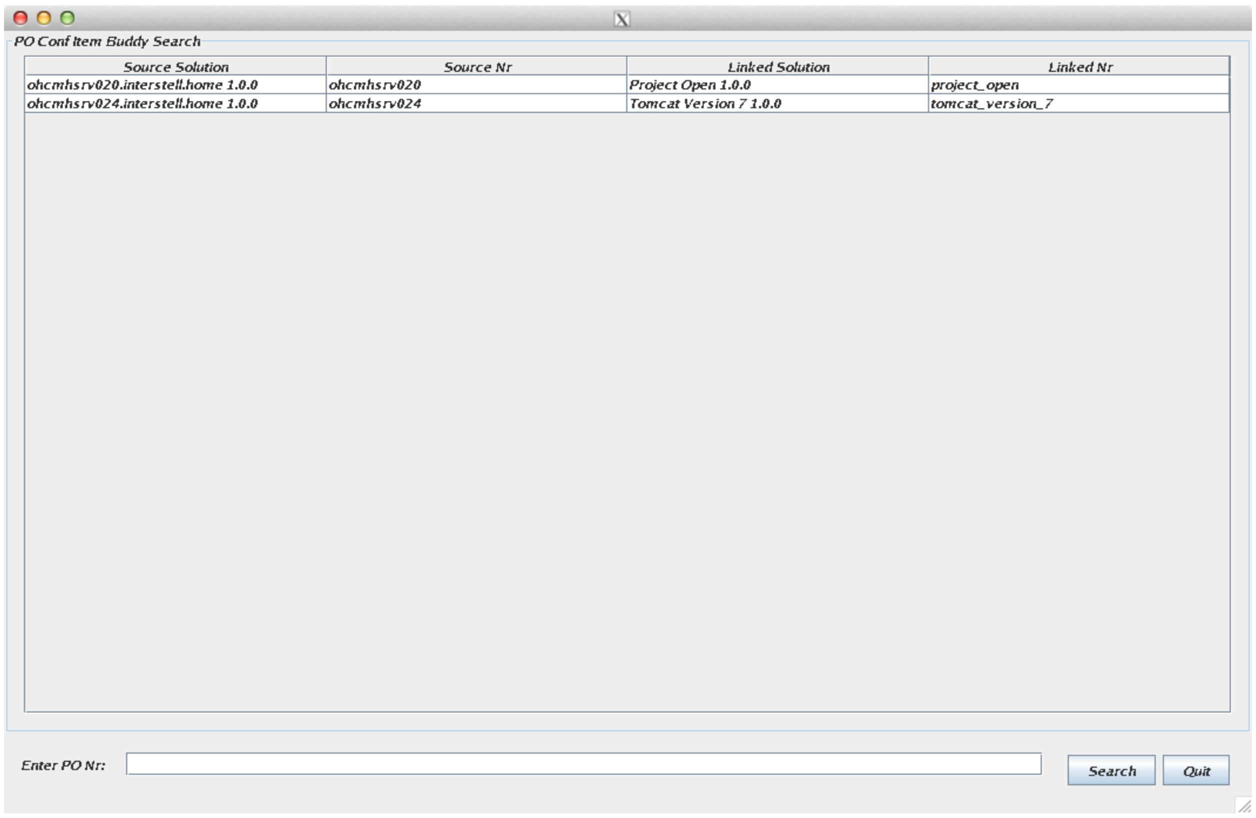
Repeat the process for "ohcmhsrv020" and "project_open."

Like the meta sever relationship screen, pressing save will first verify that both Nr values are present. It'll also check to make sure the relationship's not already defined. Finally, it'll add the relationship to the tables.

From the main screen, you should find "Hardware Servers with Software" and click on this button:

`Search`

You should see a screen that looks like Figure 7.



| Source Solution | Source Nr | Linked Solution | Linked Nr |
|---|---|---|---|
| ohcmhsrv020.interstell.home 1.0.0 | ohcmhsrv020 | Project Open 1.0.0 | project_open |
| ohcmhsrv024.interstell.home 1.0.0 | ohcmhsrv024 | Tomcat Version 7 1.0.0 | tomcat_version_7 |

*Figure 7: POConfItemBuddy can show what software is installed on what server.*

Like the previous search results screen, you could enter a server's Nr value and filter just on that server. You can also click on each column to sort by that column.

**Scenario 3: Associating Applications to Application Servers**

Looking back at our Conf Items from Figure 2, we see that we have two applications that run under Tomcat that runs under ohcmhsrv024. As you might have guessed, we'll need their Nr values. In our example, they are:

| Application | Nr |
|---|---|
| My First Tomcat App | my_first_tomcat_app |
| My Second Tomcat Application | my_second_tomcat_application |

Under "Linked Software Containers/Servers," you should click on this button:

```
Add
```

You should see a screen that looks like the one in Figure 8.

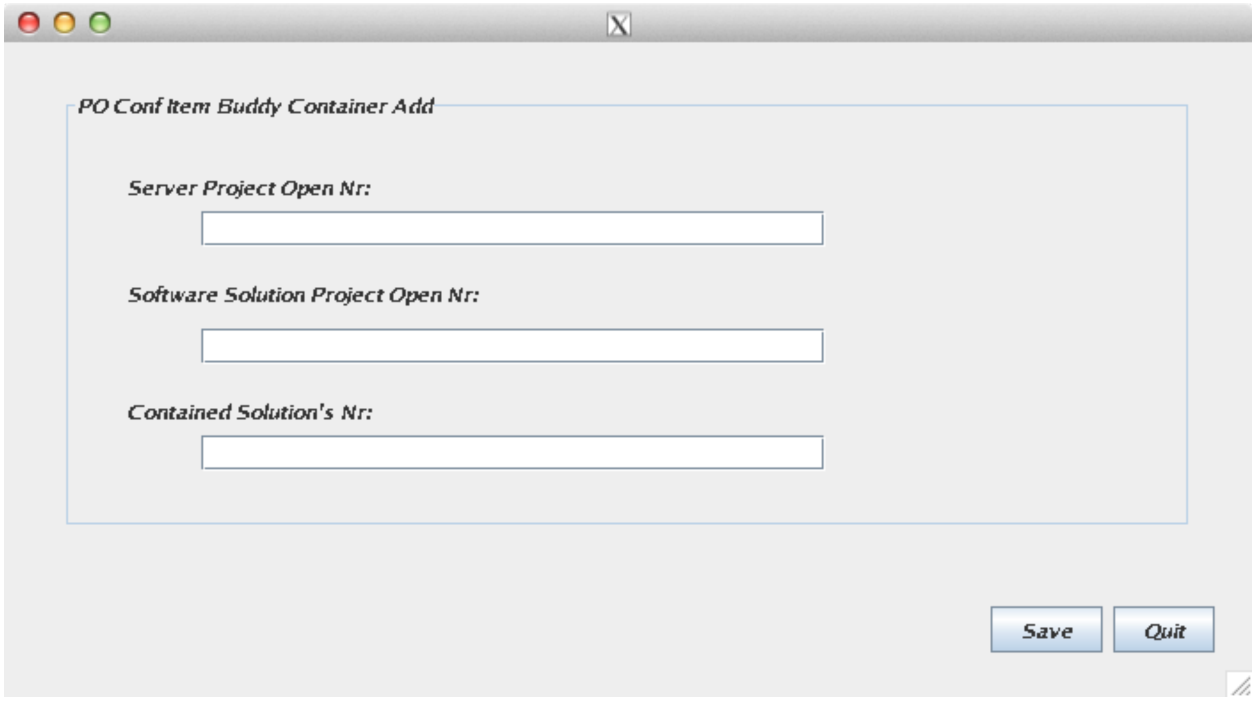

X
PO Conf Item Buddy Container Add
Server Project Open Nr:
Software Solution Project Open Nr:
Contained Solution's Nr:
Save
Quit

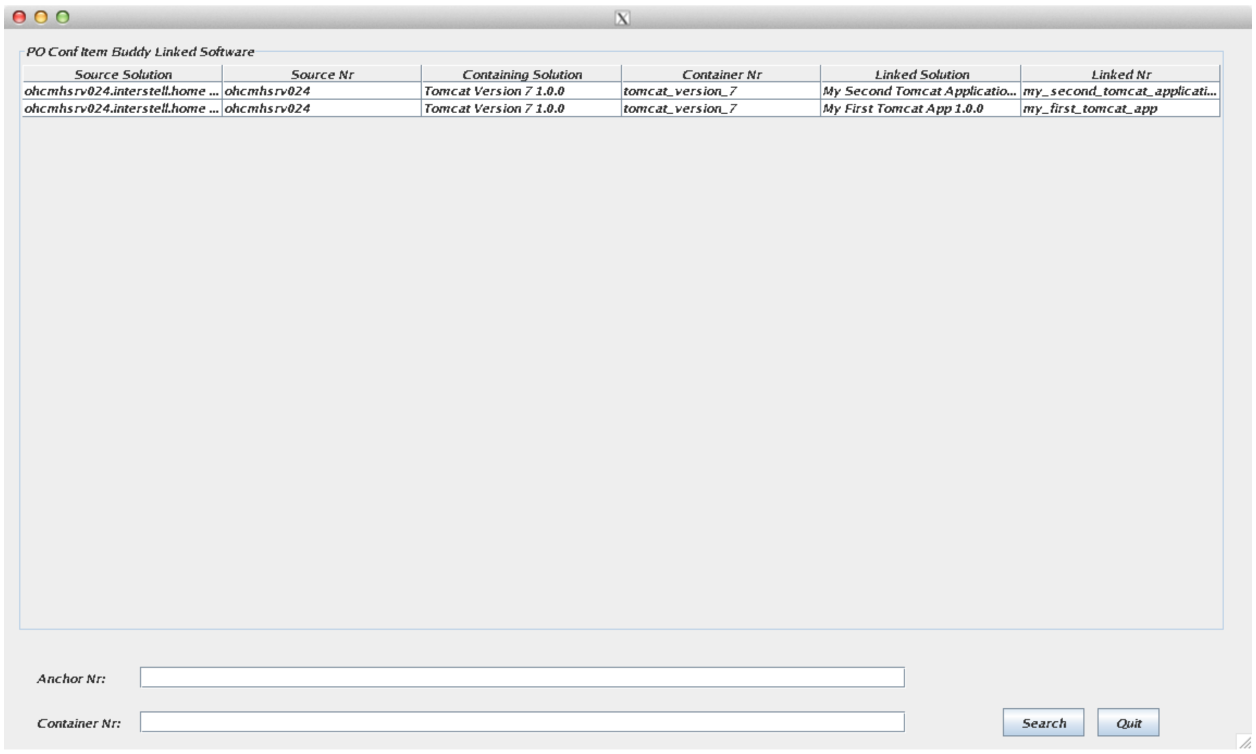
*Figure 8: POConfItemBuddy gets complicated by adding a third field! This screen can link an application to an application server running on a specific piece of hardware.*

We already know the Nr for the server (ohcmhsrv024) and the Nr for the application server (tomcat_version_7). That means we can enter those three values into the POConfItemBuddy Container Add screen.

On the main screen, find the label "Linked Software Containers/Servers." Click on this button:

```
Search
```

You should see a screen like Figure 9.



| Source Solution | Source Nr | Containing Solution | Container Nr | Linked Solution | Linked Nr |
|---|---|---|---|---|---|
| ohcmhsrv024.interstell.home ... | ohcmhsrv024 | Tomcat Version 7 1.0.0 | tomcat_version_7 | My Second Tomcat Applicatio... | my_second_tomcat_applicati... |
| ohcmhsrv024.interstell.home ... | ohcmhsrv024 | Tomcat Version 7 1.0.0 | tomcat_version_7 | My First Tomcat App 1.0.0 | my_first_tomcat_app |

*Figure 9: This screen shows what applications run under what application servers running on what servers.*

This screen offers the ability to filter by server and application server (like ohcmhsrv024 and tomcat_version_7). And of course, you can click on any of the column headings to sort by that column.

Okay, now that we've gotten some data into the system, what can we do with it?

**Scenario 4: Writing Reports**

Having all this data doesn't help much if we can't report on it. So, let's take a look at some simple reports that will help you see what kind of relationships you've defined with the POConfItemBuddy. We'll start with meta server reports.

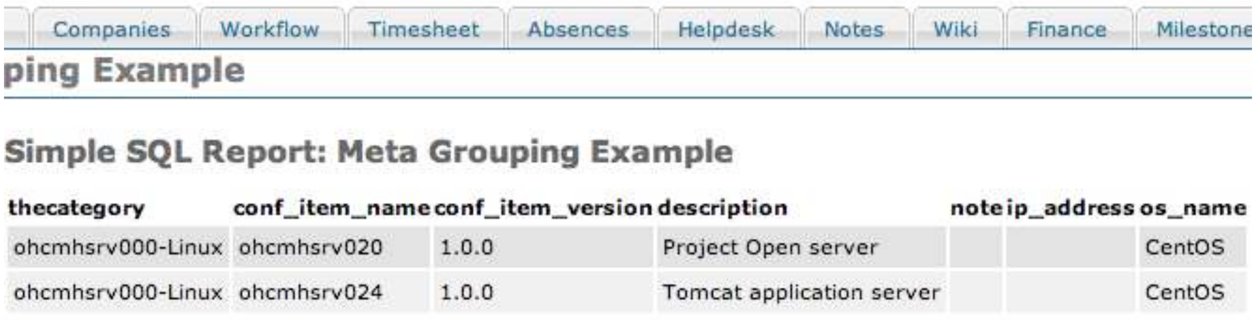
The table c1_conf_item_meta_defs includes two columns of interest:

1. ccimd_meta_nr: This is the nr of the meta item we added to ]po[; it represents a category, like All Windows Servers
2. ccimd_linked_nr: This is the nr that refers to an individual server or member of the group

A report that would pull the description of the category, plus commonly needed data for the individual members of that category, would look like this:

```
SELECT (SELECT conf_item_name FROM im_conf_items
 WHERE conf_item_nr = a.ccimd_meta_nr) AS thecategory,
 b.conf_item_name, conf_item_version, description, note, ip_address, os_name
 FROM c1_conf_item_meta_defs a, im_conf_items b
 WHERE b.conf_item_nr = a.ccimd_linked_nr
 ORDER BY thecategory, conf_item_name
```

The results might look something like Figure 10.



Simple SQL Report: Meta Grouping Example

| thecategory | conf_item_name | conf_item_version | description | note | ip_address | os_name |
|---|---|---|---|---|---|---|
| ohcmhsrv000-Linux | ohcmhsrv020 | 1.0.0 | Project Open server | | | CentOS |
| ohcmhsrv000-Linux | ohcmhsrv024 | 1.0.0 | Tomcat application server | | | CentOS |

*Figure 10: Example of a simple meta grouping report.*

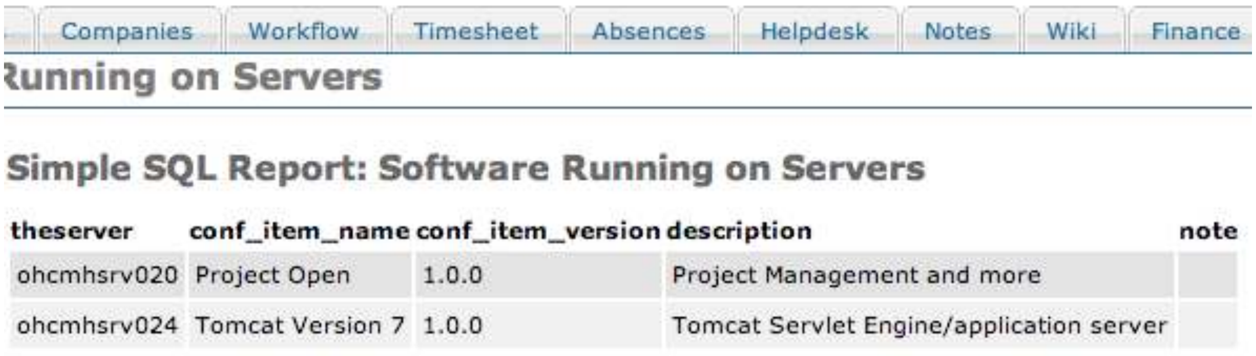
The second kind of report we can get out of ]po[ now is a report listing all of the software that's installed on a given piece of hardware. The basis for this report is c1_conf_item_linkage. This table has two important columns:

1. ccil_anchor_nr: This is the nr representing the server
2. ccil_linked_nr: This is the nr representing the software that runs on the server

A query that lists all of the software running on all of the servers might look like this:

```
SELECT (SELECT conf_item_name FROM im_conf_items WHERE
conf_item_nr = a.ccil_anchor_nr) AS theserver,
 b.conf_item_name, b.conf_item_version, description, note
 FROM c1_conf_item_linkage a, im_conf_items b
 WHERE b.conf_item_nr = a.ccil_linked_nr
 ORDER BY theserver, conf_item_name
```

That report might look like the screen in Figure 11.



Simple SQL Report: Software Running on Servers

| theserver | conf_item_name | conf_item_version | description | note |
|---|---|---|---|---|
| ohcmhsrv020 | Project Open | 1.0.0 | Project Management and more | |
| ohcmhsrv024 | Tomcat Version 7 | 1.0.0 | Tomcat Servlet Engine/application server | |

*Figure 11: Example of a report showing what software's running on what servers.*

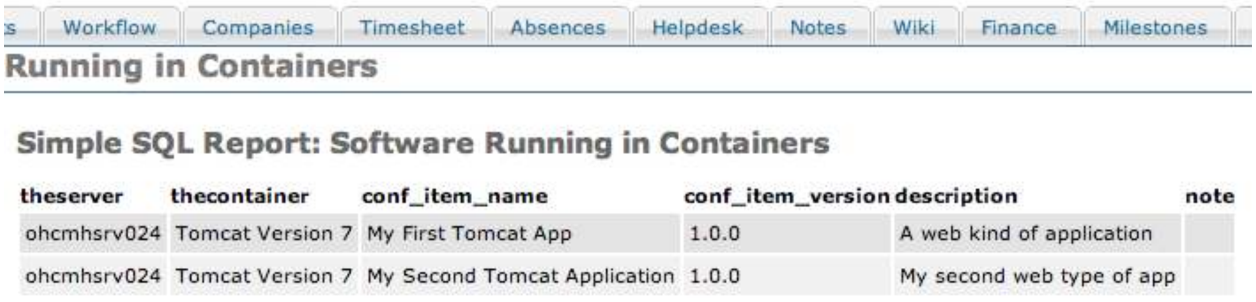
The third kind of report builds on the second in that it will display not only all of the software running on a server; it will display all of the applications running within an application container. The report that helps us perform this task is c1_conf_item_container_linkage, and it has three important columns:

1. ccicl_anchor_nr: This is nr representing the server on which things are running
2. ccicl_container_nr: This is the nr that points to the container
3. ccicl_linked_nr: This is the nr that points to the individual software

A query that shows all software running just in containers might look like this:

```
SELECT (SELECT conf_item_name FROM im_conf_items WHERE
conf_item_nr = a.ccicl_anchor_nr) AS theserver,
 (SELECT conf_item_name FROM im_conf_items WHERE conf_item_nr =
a.ccicl_container_nr) AS thecontainer,
 b.conf_item_name, conf_item_version, description, note
 FROM c1_conf_item_container_linkage a, im_conf_items b
 WHERE b.conf_item_nr = a.ccicl_linked_nr
 ORDER BY theserver, thecontainer, conf_item_name
```

That report might produce something like looks like Figure 12.



Running in Containers

Simple SQL Report: Software Running in Containers

| theserver | thecontainer | conf_item_name | conf_item_version | description | note |
|---|---|---|---|---|---|
| ohcmhsrv024 | Tomcat Version 7 | My First Tomcat App | 1.0.0 | A web kind of application | |
| ohcmhsrv024 | Tomcat Version 7 | My Second Tomcat Application | 1.0.0 | My second web type of app | |

*Figure 12: Example report showing what software's running on containers (application servers).*

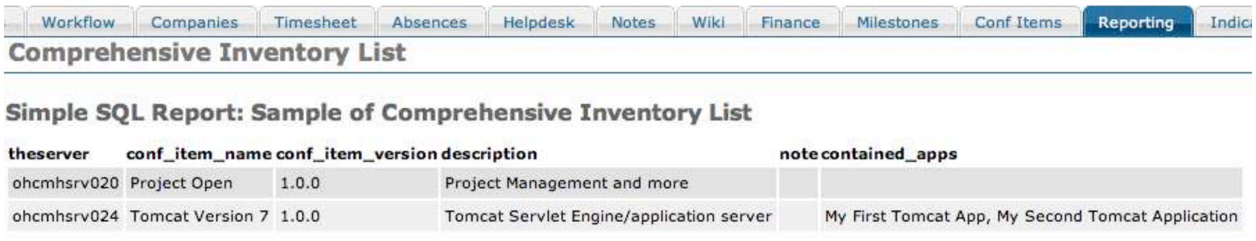
The last kind of report we can run is a comprehensive report that shows not only the software running on a server, but the software running under software containers running on the server. The code might look like this:

**Note:** In the spirit of giving credit where credit is due, I found the technique to convert multiple rows to a comma-delimited list here:

http://webcache.googleusercontent.com/search?q=cache:IrWI1LfUPtwJ:postgres.cz/wiki/PostgreSQL_SQL_Tricks+&cd=11&hl=en&ct=clnk&gl=us

```
SELECT (SELECT conf_item_name FROM im_conf_items WHERE conf_item_nr =
a.ccil_anchor_nr) AS theserver,
 b.conf_item_name, b.conf_item_version, b.description, b.note,
 ARRAY_TO_STRING(ARRAY(
      --SELECT c.ccicl_linked_nr FROM c1_conf_item_container_linkage c
      SELECT d.conf_item_name FROM c1_conf_item_container_linkage c,
im_conf_items d
      WHERE c.ccicl_container_nr = a.ccil_linked_nr
      AND d.conf_item_nr = c.ccicl_linked_nr
 ), ', ') AS contained_apps
 FROM c1_conf_item_linkage a, im_conf_items b
 WHERE b.conf_item_nr = a.ccil_linked_nr
 ORDER BY theserver, conf_item_name
```

The results of that command might look like something in Figure 13.



Comprehensive Inventory List

Simple SQL Report: Sample of Comprehensive Inventory List

| theserver | conf_item_name | conf_item_version | description | note | contained_apps |
|---|---|---|---|---|---|
| ohcmhsrv020 | Project Open | 1.0.0 | Project Management and more | | |
| ohcmhsrv024 | Tomcat Version 7 | 1.0.0 | Tomcat Servlet Engine/application server | | My First Tomcat App, My Second Tomcat Application |

*Figure 13: Example of comprehensive inventory report.*

I hope that POConfItemBuddy is useful for you. Please feel free to post questions and suggestions to the Source Forge page here:

http://sourceforge.net/p/poconfitembuddy/discussion/general/

## Appendix A: Contents of PostgreSQL Tables

Here're the scripts POConfItemBuddy uses to create its tables:

### c1_conf_item_meta_defs

```
CREATE TABLE c1_conf_item_meta_defs
(
    ccimd_meta_nr TEXT NOT NULL,
    ccimd_linked_nr TEXT NOT NULL,
    ccimd_date_entered DATE NOT NULL,
    PRIMARY KEY (ccimd_meta_nr, ccimd_linked_nr)
);

CREATE INDEX c1_conf_item_meta_defs_ndx01 ON c1_conf_item_meta_defs
(ccimd_meta_nr);
```

### c1_conf_item_linkage

```
CREATE TABLE c1_conf_item_linkage
(
    ccil_anchor_nr TEXT NOT NULL,
    ccil_linked_nr TEXT NOT NULL,
    ccil_date_entered DATE NOT NULL,
    PRIMARY KEY (ccil_anchor_nr, ccil_linked_nr)
);

CREATE INDEX c1_conf_item_linkage_ndx01 ON c1_conf_item_linkage
(ccil_anchor_nr);
```

### c1_conf_item_container_linkage

```
CREATE TABLE c1_conf_item_container_linkage
(
    ccicl_anchor_nr TEXT NOT NULL,
    ccicl_container_nr TEXT NOT NULL,
    ccicl_linked_nr TEXT NOT NULL,
    ccicl_date_entered DATE NOT NULL,
    FOREIGN KEY (ccicl_anchor_nr, ccicl_container_nr) REFERENCES
c1_conf_item_linkage(ccil_anchor_nr, ccil_linked_nr)
);

CREATE INDEX c1_conf_item_container_linkage_ndx01 ON
c1_conf_item_container_linkage (ccicl_anchor_nr);
```