Posh: an OSGi Shell
RFC132 in action!

Derek Baum

www.paremus.com/devcon2009download/

The (Draft) Specification

RFC 0132 Command Line Interface and Launching

Draft

41 Pages

Abstract

This RFC describes a proposed specification for a Command processing interface for the OSGi Framework.

RFC132: Problem Description

No standard way for user to interact with an OSGi-based system.

- Different commands for common tasks
  - install, start, stop, status
- Different API to add commands
  - Commands are not reusable
- No script support

Equinox & Felix Shells

```
$ java -jar org.eclipse.osgi_3.4.3.R34x_v20081215-1030.jar
$ java -jar org.eclipse.osgi_3.4.3.R34x_v20081215-1030.jar -console
osgi> ss
Framework is launched.
id State       Bundle
0  ACTIVE      org.eclipse.osgi_3.4.3.R34x_v20081215-1030
osgi> ^D$


$ java -jar bin/felix.jar
-> ss
Command not found.
-> ps
START LEVEL 1
   ID   State         Level  Name
[   0] [Active     ] [    0] System Bundle (1.8.0)
[   1] [Active     ] [    1] Apache Felix Shell Service (1.2.0)
[   2] [Active     ] [    1] Apache Felix Shell TUI (1.2.0)
[   3] [Active     ] [    1] Apache Felix Bundle Repository (1.4.0)
-> ^DShellTUI: No standard input...exiting.
^C$
```

Hello, Equinox

```
import org.eclipse.osgi.framework.console.CommandInterpreter;
import org.eclipse.osgi.framework.console.CommandProvider;

public class Commands implements CommandProvider {

  public void _hello(CommandInterpreter ci) {
    ci.print("Hello, " + ci.nextArgument());
  }

  public String getHelp() {
    return "\thello - say hello\n";
  }
}
```

Hello, Felix

```
import org.apache.felix.shell.Command;

public class HelloCommand implements Command {

  public void execute(String s, PrintStream out, PrintStream err) {
    String[] args = s.split("\\s+");

    if (args.length < 2) {
      err.println("Usage: " + getUsage());
    }
    else {
      out.println("Hello, " + args[1]);
    }
  }

  public String getName() { return "hello";}
  public String getUsage() { return "hello name";}
  public String getShortDescription() { return "say hello";}
}
```

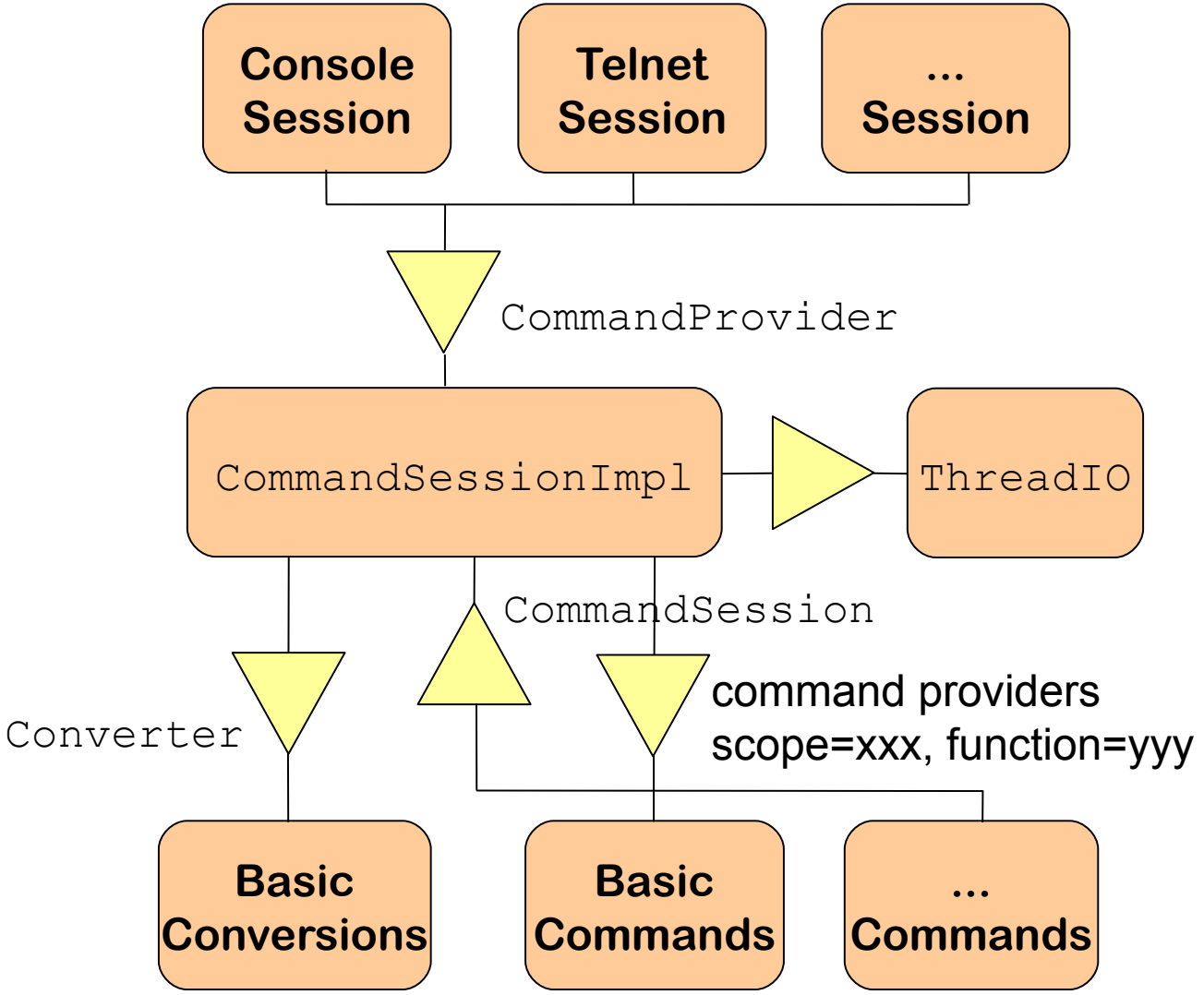
Hello, RFC132

```
public class Commands {

  public boolean hello(String name) {
    System.out.println("Hello, " + name);
    return name != null;
  }

}
```

RFC132 Command Interface

- Simple & lightweight to add commands
  - Promotes adding commands in any bundle
- Access to input and output streams
- Shared session state between commands
- Small (core runtime < 50Kb)

Architecture



CommandProcessor

```
package org.osgi.service.command;

public interface CommandProcessor {
   CommandSession createSession(InputStream in, PrintStream out,
    PrintStream err);
}
```

CommandSession

```
package org.osgi.service.command;

public interface CommandSession {
  Object execute(CharSequence program) throws Exception;
  void close();

  InputStream getKeyboard();
  PrintStream getConsole();

  Object get(String name);
  void put(String name, Object value);

  CharSequence format(Object target, int level);
  Object convert(Class<?> type, Object instance);
}
```

Command Provider

- **There is no `CommandProvider` interface**
- **Parameters are *coerced* using reflection**
- **Any service can provide commands**

```
public void start(BundleContext ctx) {
  Commands cmd = new Commands();
  String functions[] = {“hello”, “goodbye”};
  Dictionary dict = new Dictionary();
  dict.put(“osgi.command.scope”, “myscope”);
  dict.put(“osgi.command.function”, functions);
  ctx.registerService(cmd.getClass().getName,() cmd, dict);
}
```

Any Command Interface

```
public boolean grep(String[] args) throws IOException;


public URI cd(CommandSession session, String dir);
public URI pwd(CommandSession session);


public void addCommand(String scope, Object target);
public void addCommand(String scope, Object target, Class<?> fc);
public void addCommand(String scope, Object target, String func);


public Bundle getBundle(long id);
public Bundle[] getBundles();
public ServiceRegistration registerService(
        String clazz, Object svcObj, Dictionary dict);
```

Grep command

```
public boolean grep(CommandSession session, List<String> args)
                                                  throws IOException {
  Pattern pattern = Pattern.compile(args.remove(0));

  for (String arg : args) {
    InputStream in =
              Directory.resolve(session,arg).toURL().openStream();
    Reader rdr = new BufferedReader(new InputStreamReader(in));
    String s;

    while ((s = rdr.readLine()) != null) {
      if (pattern.matcher(s).find()) {
        match = true;
        System.out.println(s);
      }
    }
  }

  return match;
}
```

Interactive Shell Language

- Interactive shell requirements differ from those for scripting languages
  - Easy for users to type
  - Minimum parenthesis, semi-colons etc
  - Compatibility with Unix shells like bash
- Existing JVM languages not good for shell
  - Jacl: needs own TCL-derived library
  - BeanShell: syntax too verbose

Tiny Shell Language - TSL

- Easy to use – no unnecessary syntax
- Provides lists, pipes and closures
- Leverages Java capabilities
  - Method calls using reflection
  - Argument coercion
- Commands can provide control primitives
- Small size (<50Kb)

Closures implement Function

```
package org.osgi.service.command;
public interface Function {
  Object execute(CommandSession session, List<Object> arguments)
   throws Exception;
}
```

Add Control Primitives

```
public void each(CommandSession session, Collection<Object> list,
   Function closure) throws Exception {

  List<Object> args = new ArrayList<Object>();
  args.add(null);

  for (Object x : list) {
    args.set(0, x);
    closure.execute(session, args);
  }
}
```

```
% each [a, b, 1, 2] { echo $it }
a
b
1
2
```

RFC132: What's missing?

- Commands in osgi: scope are not defined
- Search order for unscoped commands
- List available commands and variables
- Script execution
- Command editing, history and completion

Posh: in action!

Posh in action!

Builds on basic RFC132 runtime

- Adds SCOPE path
- Shell builtins: set, type
- Script execution
- Command editing, history & completion
- Control-C interrupt handling
- And more…

Paremus

Links

RFC132 Draft Specification
www.osgi.org/download/osgi-4.2-early-draft.pdf

Posh "devcon" binary download
www.paremus.com/devcon2009download/

derek.baum@paremus.com