

N S 2022

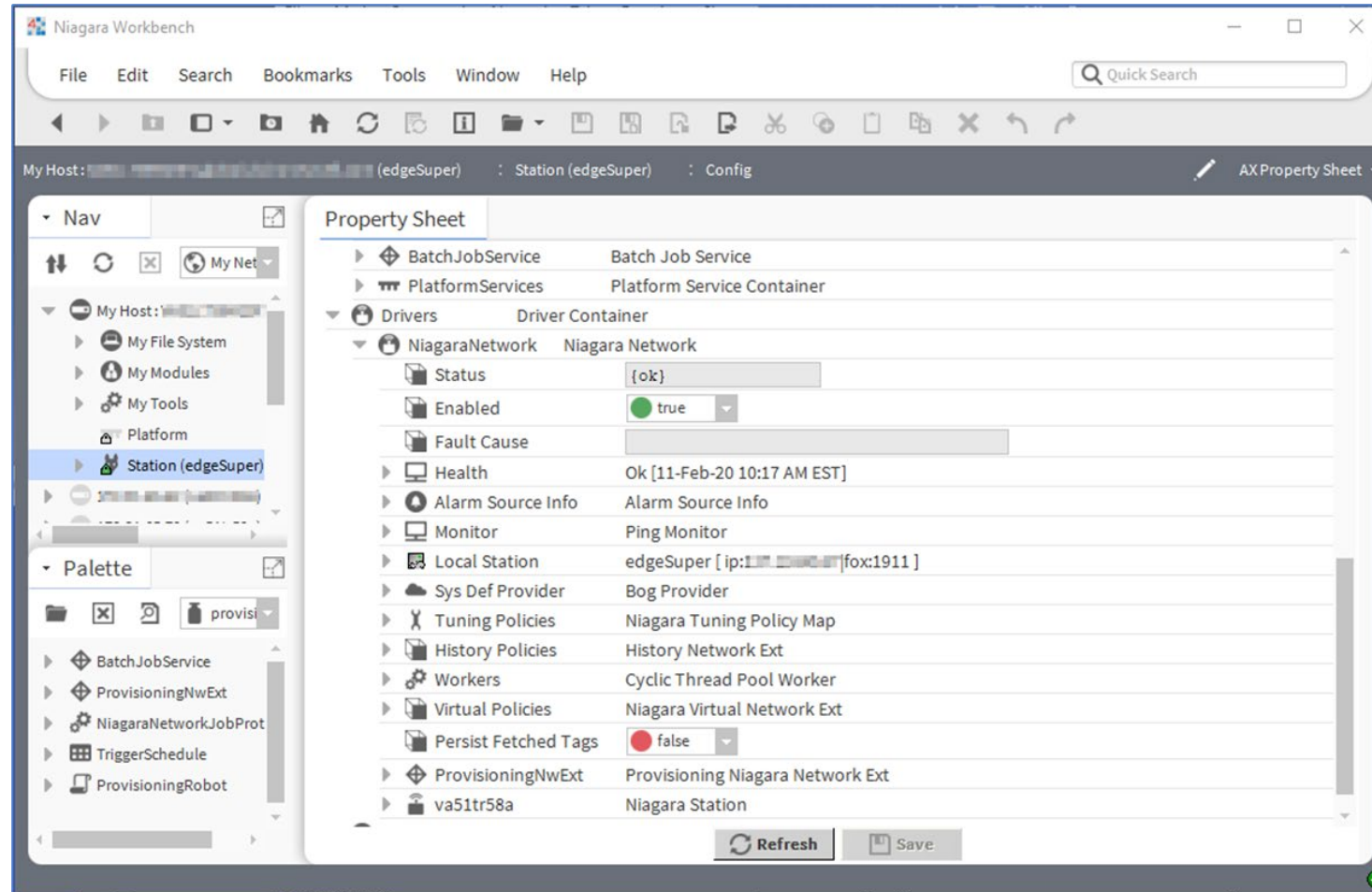
ACCELERATING INNOVATION

Developing Custom Provisioning Steps

Tim Urenda
Tridium, Inc.

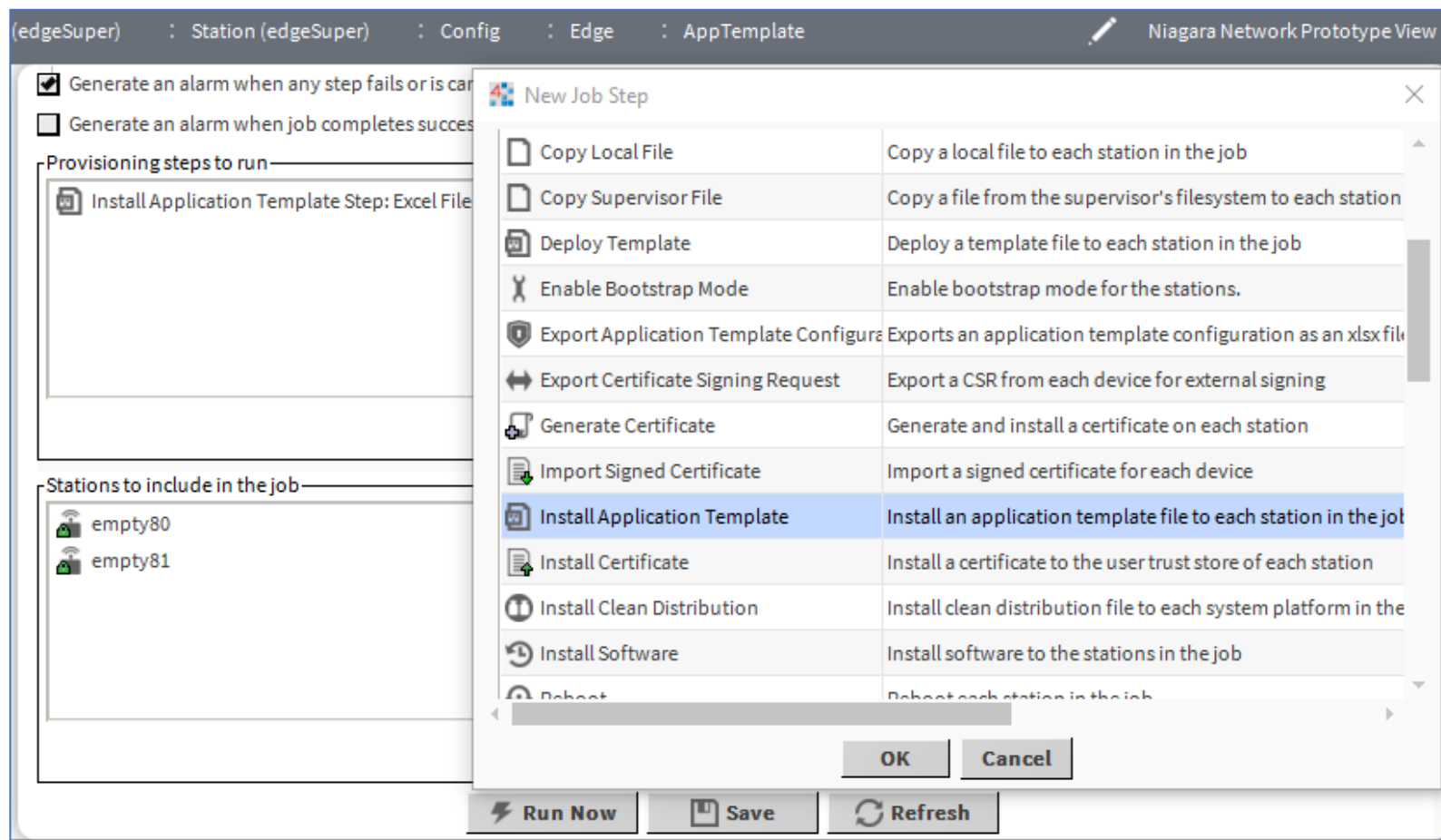


What is Provisioning?



Provisioning Setup on a Supervisor

- Backups
- Software
- Templates
- Credentials
- Certificates
- Licenses
- Etc.



Provisioning Public API

- batchJob-rt
 - BBatchJob
 - BBatchJobService
 - BJobStep
 - BDeviceStepDetails
- provisioningNiagara-wb
 - BStationStepFactory
- batchJob-wb

The screenshot shows the Niagara Help application interface. The left sidebar displays a tree view of the API structure, with the package `javax.baja.batchJob.driver` expanded and `BDeviceJobStep` selected. The main content area displays the details for the `public abstract class BDeviceJobStep`. The details include the module part (`batchJob-rt`), package (`javax.baja.batchJob.driver`), and the class it extends (`BJobStep`). A description states: "Base class for a job step that is to be run for each device in a BForEachDeviceStage. The step creates a BDeviceStepDetails for each device it processes during stage execution." The "Since" section indicates it is available in "Baja 1.0". Below this, there are sections for "Subclasses" (listing `BDeviceJobStep`), "Constructors" (listing `public BDeviceJobStep()`), and "Methods" (listing `public boolean canCombine(BDeviceJobStep step)`, `protected void checkCanceled(BDevice device)`, and `public void combine(BDeviceJobStep step)`).

Extending Provisioning

- Our goal: Install an ACE application on EDGE devices
- Create a new module
- Load it into the IDE
- Create classes

New Module Wizard
Step 1 of 3

Create Module under Directory
/D:/niagara/r413

Module Name aceAppProv
Preferred Symbol aap
Version 1.0
Description Provision an ACE application file to EDGE devices
Vendor ACME

Runtime Profiles

- RUNTIME: Module JARs having core runtime Java classes only, no user interface.
- UX: Module JARs having lightweight HTML5+JavaScript+CSS user interface only.
- WB: Module JARs having Workbench or Workbench Applet user interface classes.
- SE: Module JARs having Java classes that use the full Java 8 Standard Edition (SE) platform API.

Create Lexicon
 Create Palette

Back Next Finish Cancel

Create Classes

- BAceAppInstallStep
- BAceAppInstallFactory
- BAceAppInstallTest
- Niagara annotations
- Base classes
- Method overrides
- Slotomatic

```
public BJobStep makeStep(BWidget owner, BBatchJob batchJob, BObject jobTarget, BObject source, Context context)
    throws Exception
{
    // Prompt for the ACE file
    BFileChooser chooser = BFileChooser.makeOpen(owner);
    chooser.setCurrentDirectory(BOrd.make("local:|file::~ace"));
    chooser.clearFilters();
    chooser.addFilter(new ExtFileFilter(BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.fileFilter"), "ace"));
    chooser.setUpdateFilenameExtensionFromFilter(true);
    BOrd fileOrd = chooser.show();
    if (fileOrd == null)
    {
        // The file chooser was canceled, so bail
        return null;
    }

    // Display warning that running this job will overwrite any existing ACE application
    if (BDialog.CANCEL == BDialog.open(owner, BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.warningTitle"),
        BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.deployWarning") + "\n\n" +
        BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.backupWarning"),
        BDialog.OK_CANCEL, BDialog.WARNING_ICON, (String)null))
    {
        return null;
    }

    // Copy the ACE file to the supervisor.
    try
    {
        BFile file = (BFile)fileOrd.resolve().get();
        BObject cacheDir = getTargetDir(BOrd.make(BAceAppInstallStep.ACE_CACHE), jobTarget);
        if (!deleteExistingFile(file.getFileName(), (BDirectory)cacheDir, owner))
        {
            return null;
        }

        Mark mark = new Mark((BObject)file, file.getFileName());
        mark.copyTo(cacheDir, new BComponent(), new BasicContext());

        // Show confirmation of file copy
        BDialog.info(owner,
            BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.copyAceFile.complete"),
            BString.make(BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.copyAceFile.completeDetail"),
                file.getFileName(), BAceAppInstallStep.ACE_CACHE));

        // Create the job step
        BAceAppInstallStep step = new BAceAppInstallStep();
        step.setTargetFile(file.getFileName());
    }
}
```



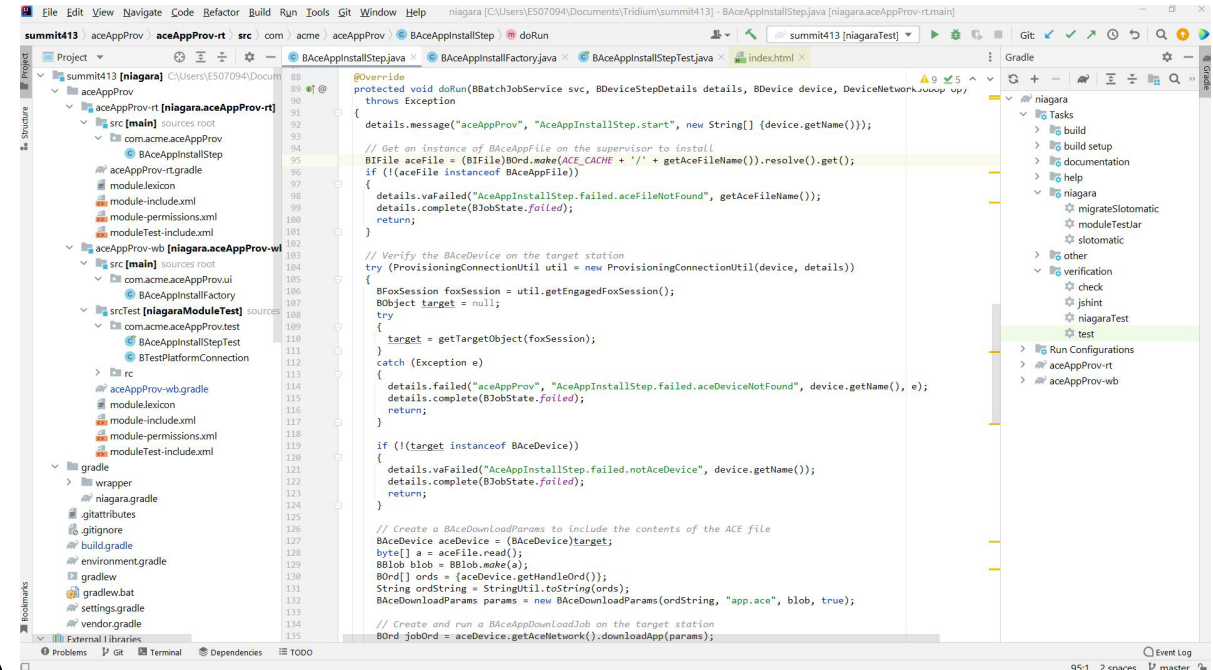






Provisioning Step

- ACE application file on supervisor
- ACE Device component on target station
 - ProvisioningConnectionUtil
- ACE application install
 - Re-use methods available to Workbench



```
protected void doRun(BBatchJobService svc, BDeviceStepDetails details, BDevice device, DeviceNetworkJob job) throws Exception {
    details.message("aceAppProv", "AceAppInstallStep.start", new String[] {device.getName()});
    // Get an instance of BACEAppFile on the supervisor to install
    BFile aceFile = (BFile)BOrd.make(ACE_CACHE + '/' + getAceFileName()).resolve().get();
    if (!(aceFile instanceof BACEAppFile)) {
        details.vFailed("AceAppInstallStep.failed.aceFileNotFound", getAceFileName());
        details.complete(BJobState.failed);
        return;
    }
    // Verify the BACEDevice on the target station
    try (ProvisioningConnectionUtil util = new ProvisioningConnectionUtil(device, details)) {
        BFoxSession foxSession = util.getEngagedFoxSession();
        BObject target = null;
        try {
            target = getTargetObject(foxSession);
        } catch (Exception e) {
            details.failed("aceAppProv", "AceAppInstallStep.failed.aceDeviceNotFound", device.getName(), e);
            details.complete(BJobState.failed);
            return;
        }
        if (!(target instanceof BACEDevice)) {
            details.vFailed("AceAppInstallStep.failed.notAceDevice", device.getName());
            details.complete(BJobState.failed);
            return;
        }
        // Create a BACEDownloadParams to include the contents of the ACE file
        BACEDevice aceDevice = (BACEDevice)target;
        byte[] a = aceFile.read();
        BBlob blob = BBlob.make(a);
        BOrd[] ords = {aceDevice.getHandleOrd()};
        String ordString = StringUtil.toString(ords);
        BACEDownloadParams params = new BACEDownloadParams(ordString, "app.ace", blob, true);
        // Create and run a BACEAppDownloadJob on the target station
        BOrd jobOrd = aceDevice.getAceNetwork().downloadApp(params);
    }
}
```

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection

```

88 @Override
89 protected void doRun(BBatchJobService svc, BDeviceStepDetails details, BDevice device, DeviceNetworkJobUp,
90 throws Exception
91 {
92     details.message("aceAppProv", "AceAppInstallStep.start", new String[] {device.getName()});
93
94     // Get an instance of BAceAppFile on the supervisor to install
95     BIFile aceFile = (BIFile)BOrd.make(ACE_CACHE + '/' + getAceFileName()).resolve().get();
96     if (!(aceFile instanceof BAceAppFile))
97     {
98         details.vaFailed("AceAppInstallStep.failed.aceFileNotFound", getAceFileName());
99         details.complete(BJobState.failed);
100         return;
101     }
102
103     // Verify the BAceDevice on the target station
104     try (ProvisioningConnectionUtil util = new ProvisioningConnectionUtil(device, details))
105     {
106         BFoxSession foxSession = util.getEngagedFoxSession();
107         BObject target = null;
108         try
109         {
110             target = getTargetObject(foxSession);
111         }
112         catch (Exception e)
113         {
114             details.failed("aceAppProv", "AceAppInstallStep.failed.aceDeviceNotFound", device.getName(), e);
115             details.complete(BJobState.failed);
116             return;
117         }
118
119         if (!(target instanceof BAceDevice))
120         {
121             details.vaFailed("AceAppInstallStep.failed.notAceDevice", device.getName());
122             details.complete(BJobState.failed);
123             return;
124         }
125
126         // Create a BAceDownloadParams to include the contents of the ACE file
127         BAceDevice aceDevice = (BAceDevice)target;
128         byte[] a = aceFile.read();
129         BBlob blob = BBlob.make(a);
130         BOrd[] ords = {aceDevice.getHandleOrd()};
131         String ordString = StringUtil.toString(ords);
132         BAceDownloadParams params = new BAceDownloadParams(ordString, "app.ace", blob, true);
133
134         // Create and run a BAceAppDownloadJob on the target station
135         BOrd jobOrd = aceDevice.getAceNetwork().downloadApp(params);

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection

```

126 // Create a BAceDownloadParams to include the contents of the ACE file
127 BAceDevice aceDevice = (BAceDevice)target;
128 byte[] a = aceFile.read();
129 BBlob blob = BBlob.make(a);
130 BOrd[] ords = {aceDevice.getHandleOrd()};
131 String ordString = StringUtil.toString(ords);
132 BAceDownloadParams params = new BAceDownloadParams(ordString, "app.ace", blob, true);
133
134 // Create and run a BAceAppDownloadJob on the target station
135 BOrd jobOrd = aceDevice.getAceNetwork().downloadApp(params);
136 BAceAppDownloadJob job = null;
137 try
138 {
139     aceDevice.getComponentSpace().sync();
140     job = (BAceAppDownloadJob)jobOrd.get(aceDevice);
141 }
142 catch(Exception e)
143 {
144     details.vaadinFailed("AceAppInstallStep.failed.installJobStartFailed", device.getName());
145     details.complete(BJobState.failed);
146     return;
147 }
148 // Lease the job on the target station so we can get the status as it runs
149 job.lease();
150 details.message("aceAppProv", "AceAppInstallStep.downloadRunning",
151     new String[] {jobOrd.encodeToString(), job.getJobState().getTag()});
152 waitForJobComplete(job);
153
154 // Check if any modules are missing or other failure occurred
155 if(job.getJobState() != BJobState.success)
156 {
157     details.message("aceAppProv", "AceAppInstallStep.failed.jobState", job.getJobState().getTag());
158
159     // Job failed, first check if there are missing modules
160     BString missingModules = job.getMissingModules();
161     if(missingModules != null)
162     {
163         details.vaadinFailed("AceAppInstallStep.failed.missingModule", device.getName(), missingModules.encodeToString());
164         details.complete(BJobState.failed);
165         return;
166     }
167
168     // Otherwise return generic failure message
169     details.vaadinFailed("AceAppInstallStep.failed.installFailed", device.getName());
170     details.complete(BJobState.failed);
171     return;
172 }
173

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection

```

171         return;
172     }
173
174     // We got here, job was successful
175     details.success();
176     details.complete(BJobState.success);
177
178 }
179 catch (Exception e)
180 {
181     details.failed("provisioningNiagara", "AceAppInstallStep.failed.exception", device.getName(), e);
182     details.complete(BJobState.failed);
183 }
184 return;
185 }
186
187 // This code is separated into a method to make testing easier
188 public BObject getTargetObject(BFoxSession foxSession)
189 {
190     // Use the Fox session as the root for resolving station ORDs
191     BOrd targetRootOrd = BOrd.make("station:|slot:/Drivers");
192     BComponent targetRoot = (BComponent)targetRootOrd.get(foxSession);
193     targetRoot.lease(2);
194     BComponent[] aceDeviceArray = CompUtil.getDescendants(targetRoot, BAceDevice.class);
195     return Arrays.asList(aceDeviceArray).stream().findAny().get();
196 }
197
198 // Poll to check if the BAceAppDownloadJob is complete
199 public void waitForJobComplete(BAceAppDownloadJob job)
200 {
201     int count = 0;
202     while(!job.getJobState().isComplete() && count<240)
203     {
204         DrUtil.wait(500);
205         count++;
206     }
207 }
208
209 @Override
210 public String toString(Context cx)
211 {
212     return Lexicon.getText("AceAppInstallStep.display", getAceFileName());
213 }
214
215 public static final Lexicon lexicon = Lexicon.make("aceAppProv");
216 public static final String ACE_CACHE = "file:^aceFileCache";
217 }
218

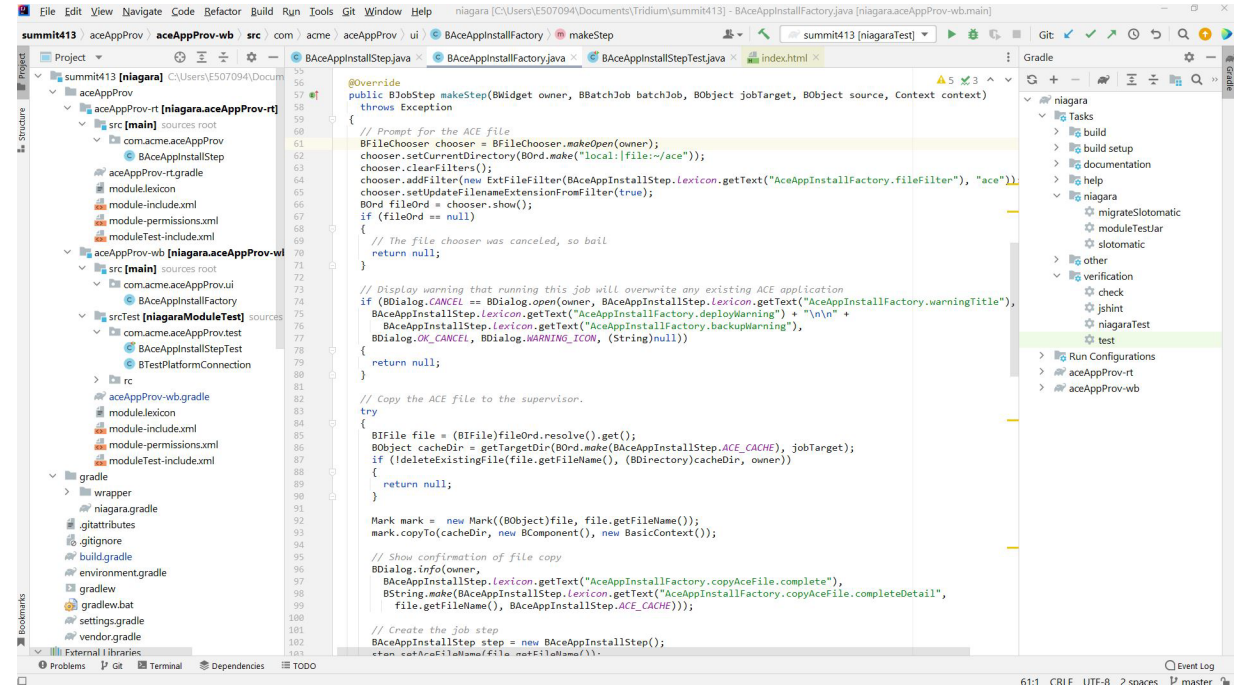
```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Step Factory

- Select ACE file for installation
- Copy file to supervisor
- Create step component
- Dialogs for user feedback



```
public BJobStep makeStep(BWidget owner, BBatchJob batchJob, BObject jobTarget, BObject source, Context context) throws Exception {  
    // Prompt for the ACE file  
    BFileChooser chooser = BFileChooser.makeOpen(owner);  
    chooser.setCurrentDirectory(BOrd.make("local:file::~ace"));  
    chooser.clearFilters();  
    chooser.addFilter(new ExtFileFilter(BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.fileFilter"), "ace"));  
    chooser.setUpdateFileNameExtensionFromFilter(true);  
    BOrd fileOrd = chooser.show();  
    if (fileOrd == null) {  
        // The file chooser was canceled, so bail  
        return null;  
    }  
    // Display warning that running this job will overwrite any existing ACE application  
    if (BDialog.CANCEL == BDialog.open(owner, BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.warningTitle"),  
        BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.deployWarning") + "\n\n" +  
        BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.backupWarning"),  
        BDialog.OK_CANCEL, BDialog.WARNING_ICON, (String)null)) {  
        return null;  
    }  
    // Copy the ACE file to the supervisor.  
    try {  
        BFile file = (BFile)fileOrd.resolve().get();  
        BObject cacheDir = getTargetDir(BOrd.make(BAceAppInstallStep.ACE_CACHE), jobTarget);  
        if (!deleteExistingFile(file.getFileName(), (BDirectory)cacheDir, owner)) {  
            return null;  
        }  
        Mark mark = new Mark((BObject)file, file.getFileName());  
        mark.copyTo(cacheDir, new BComponent(), new BasicContext());  
        // Show confirmation of file copy  
        BDialog.info(owner,  
            BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.copyAceFile.complete"),  
            BString.make(BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.copyAceFile.completeDetail"),  
                file.getFileName(), BAceAppInstallStep.ACE_CACHE));  
        // Create the job step  
        BAceAppInstallStep step = new BAceAppInstallStep();  
        return step.setFileName(file.getFileName());  
    }  
}
```


Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection

```

55
56
57 @Override
58 public BJobStep makeStep(BWidget owner, BBatchJob batchJob, BObject jobTarget, BObject source, Context context)
59     throws Exception
60 {
61     // Prompt for the ACE file
62     BFileChooser chooser = BFileChooser.makeOpen(owner);
63     chooser.setCurrentDirectory(BOrd.make("local:|file:~/ace"));
64     chooser.clearFilters();
65     chooser.addFilter(new ExtFileFilter(BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.fileFilter"), "ace"));
66     chooser.setUpdateFilenameExtensionFromFilter(true);
67     BOrd fileOrd = chooser.show();
68     if (fileOrd == null)
69     {
70         // The file chooser was canceled, so bail
71         return null;
72     }
73
74     // Display warning that running this job will overwrite any existing ACE application
75     if (BDialog.CANCEL == BDialog.open(owner, BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.warningTitle"),
76         BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.deployWarning") + "\n\n" +
77         BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.backupWarning"),
78         BDialog.OK_CANCEL, BDialog.WARNING_ICON, (String)null))
79     {
80         return null;
81     }
82
83     // Copy the ACE file to the supervisor.
84     try
85     {
86         BIFile file = (BIFile)fileOrd.resolve().get();
87         BObject cacheDir = getTargetDir(BOrd.make(BAceAppInstallStep.ACE_CACHE), jobTarget);
88         if (!deleteExistingFile(file.getFileName(), (BDirectory)cacheDir, owner))
89         {
90             return null;
91         }
92
93         Mark mark = new Mark((BObject)file, file.getFileName());
94         mark.copyTo(cacheDir, new BComponent(), new BasicContext());
95
96         // Show confirmation of file copy
97         BDialog.info(owner,
98             BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.copyAceFile.complete"),
99             BString.make(BAceAppInstallStep.Lexicon.getText("AceAppInstallFactory.copyAceFile.completeDetail",
100                 file.getFileName(), BAceAppInstallStep.ACE_CACHE)));
101
102         // Create the job step
103         BAceAppInstallStep step = new BAceAppInstallStep();
104         step.setAceFileName(file.getFileName());

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection
- gradle
 - wrapper
 - niagara.gradle
 - .gitattributes
 - .gitignore
 - build.gradle
 - environment.gradle
 - gradlew
 - gradlew.bat
 - settings.gradle
 - vendor.gradle
- External Libraries

```

76 BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.backupWarning"),
77 BDialog.OK_CANCEL, BDialog.WARNING_ICON, (String)null))
78 {
79     return null;
80 }
81
82 // Copy the ACE file to the supervisor.
83 try
84 {
85     BIFile file = (BIFile)fileOrd.resolve().get();
86     BObject cacheDir = getTargetDir(BOrd.make(BAceAppInstallStep.ACE_CACHE), jobTarget);
87     if (!deleteExistingFile(file.getFileName(), (BDirectory)cacheDir, owner))
88     {
89         return null;
90     }
91
92     Mark mark = new Mark((BObject)file, file.getFileName());
93     mark.copyTo(cacheDir, new BComponent(), new BasicContext());
94
95     // Show confirmation of file copy
96     BDialog.info(owner,
97         BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.copyAceFile.complete"),
98         BString.make(BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.copyAceFile.completeDetail",
99             file.getFileName(), BAceAppInstallStep.ACE_CACHE)));
100
101     // Create the job step
102     BAceAppInstallStep step = new BAceAppInstallStep();
103     step.setAceFileName(file.getFileName());
104     return step;
105 }
106 catch (Exception e)
107 {
108     BDialog.error(owner,
109         BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.copyAceFile.error"),
110         BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.copyAceFile.errorDetail",
111             e));
112 }
113
114 // Something went wrong if we get here
115 return null;
116 }
117
118 // Delete the ACE file if it exists, prompting the user to confirm the file overwrite
119 @private boolean deleteExistingFile(String filenameToDelete, BDirectory cacheDirectory, BWidget owner)
120 {
121     // Check for an existing file of the same name
122     BIFile fileToDelete = null;
123     BIFile[] cacheFiles = cacheDirectory.listFiles();
124     for (BIFile file : cacheFiles)

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection

```

117
118
119 // Delete the ACE file if it exists, prompting the user to confirm the file overwrite
120 private boolean deleteExistingFile(String filenameToDelete, BDirectory cacheDirectory, BWidget owner)
121 {
122     // Check for an existing file of the same name
123     BIFile fileToDelete = null;
124     BIFile[] cacheFiles = cacheDirectory.listFiles();
125     for (BIFile file : cacheFiles)
126     {
127         if (file.getFileName().equalsIgnoreCase(filenameToDelete))
128         {
129             fileToDelete = file;
130             break;
131         }
132     }
133
134     if (fileToDelete != null)
135     {
136         // Allow the user to confirm that this file will overwrite the existing cached file on the supervisor
137         if (BDialog.CANCEL == BDialog.open(owner,
138             BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.deleteExisting.title"),
139             BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.deleteExisting.message", filenameToDelete),
140             BDialog.OK_CANCEL,
141             BDialog.WARNING_ICON,
142             (String)null))
143         {
144             return false;
145         }
146
147         // Delete the file
148         try
149         {
150             fileToDelete.delete();
151         }
152         catch (Exception e)
153         {
154             BDialog.error(owner,
155                 BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.deleteExisting.title"),
156                 BAceAppInstallStep.lexicon.getText("AceAppInstallFactory.deleteExisting.error", filenameToDelete),
157                 e);
158             return false;
159         }
160     }
161
162     return true;
163 }
164
165 // Resolve a directory on the target station, creating any missing folders in the path

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection
- gradle
 - wrapper
 - niagara.gradle
 - .gitattributes
 - .gitignore
 - build.gradle
 - environment.gradle
 - gradlew
 - gradlew.bat
 - settings.gradle
 - vendor.gradle

```

156         e);
157         return false;
158     }
159 }
160
161     return true;
162 }
163
164 @
165 private BObject getTargetDir(BOrd ord, BObject target)
166     throws Exception
167 {
168     BObject obj;
169     try
170     {
171         // Attempt to resolve target ord
172         obj = ord.resolve(target).get();
173     }
174     catch(UnresolvedException e)
175     {
176         // Attempt to create any missing file directories
177         OrdQuery[] oqs = ord.parse();
178         for (OrdQuery oq : oqs)
179         {
180             if (!(oq instanceof FilePath))
181             {
182                 continue;
183             }
184             FilePath fp = (FilePath)oq;
185             BDirectory rootDir = (BDirectory)BOrd.make("file:^").resolve(target).get();
186             rootDir.getFileSpace().mkdir(fp);
187         }
188
189         // Try again
190         obj = ord.resolve(target).get();
191     }
192     return obj;
193 }
194
195 }
196
197

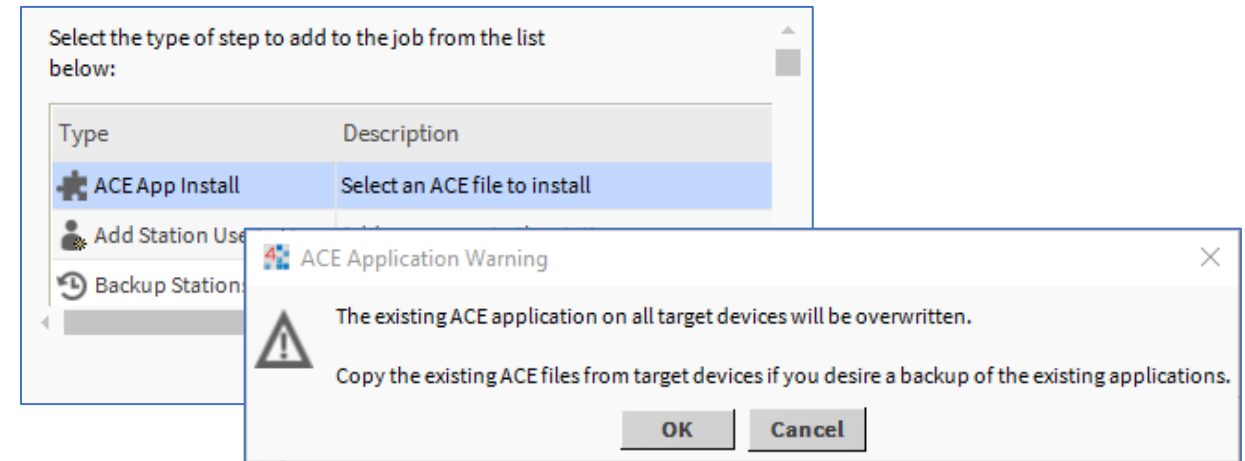
```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Lexicon and Logging

- Display information during setup
- Batch job execution messages
- Parameterized lexicon entries



Device	empty80
Job Step	Install ACE application file app.ace.
Started	28-Feb-22 1:05 PM UTC-05:00
Ended	28-Feb-22 1:05 PM UTC-05:00
State	Success

Status	Timestamp	Message
Message	28-Feb-22 1:05 PM EST	ACE application installation started on target device empty80.
Message	28-Feb-22 1:05 PM EST	ACE application installation job slot:/Services/JobService/AceAppDownloadJob3 state is running.
Success	28-Feb-22 1:05 PM EST	Step successfully completed for empty80

Compile and Run

- Add missing dependencies
- Compile jar files
 - Signing not covered here
- Configure and run the job
- View job results
 - Log files
 - History records



Test Methods

- Add classes to WB part
 - setupStation()
 - beforeMethod()
 - runStep(BDeviceJobStep)
 - makeDaemonSession(...)
- Attach test artifacts
- Mockito
- test aceAppProv

The screenshot shows an IDE window titled 'BAceAppInstallStepTest.java'. The code includes two test methods: `testAceAppInstallToEmptyStation()` and `testAceAppInstallToEmptyStation()`. The first method is annotated with `@Test` and `throws Exception`. It creates a `BAceAppInstallStep` object, sets its `aceFileName`, and calls `runStep`. It then asserts that the state is `failed` and that the job messages contain the expected text. The second method is also annotated with `@Test` and `dependsOnMethods = "testAceAppInstallToEmptyStation"`. It sets up the test environment, including `BComponent`, `BComponent drivers`, `BDevice`, and `aceDevice`.

Overlaid on the code is a test results window titled 'All suites'. The window shows the following information:

- Info**
 - [unset file name]
 - 1 test
 - 0 groups
 - Times
 - Reporter output
 - Ignored methods
 - Chronological view
- Results**
 - 2 methods, 2 passed
 - Passed methods (hide)
 - ✓ testAceAppInstall
 - ✓ testAceAppInstallToEmptyStation

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-rt.gradle
 - module.lexicon
 - module-include.xml
 - module-permissions.xml
 - moduleTest-include.xml
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection
 - rc
 - app.ace
 - BasicAceApp.ace
 - catalog_test.json
 - MinimalAceApp.ace
 - SimpleAceApp.ace
 - gradle
 - wrapper
 - niagara.gradle
 - .gitattributes
 - .gitignore
 - build.gradle
 - environment.gradle

```

90  @Override
91  @BeforeTest(alwaysRun=true,description = "Setup and start test station")
92  public void setupStation() throws Exception
93  {
94      super.setupStation();
95      // Set up the supervisor station user
96      provisioningUser = new BUser();
97      provisioningUser.setRoles(TEST_SUPER_USER);
98      BPasswordAuthenticator authenticator = new BPasswordAuthenticator();
99      authenticator.setPassword(BPassword.make("Password10"));
100     provisioningUser.setAuthenticator(authenticator);
101     getUserService().add("provisioning", provisioningUser);
102
103     // Add a provisioning network extension
104     BStation station = stationHandler.getStation();
105     BNiagaraNetwork niagaraNetwork = (BNiagaraNetwork) station.get("Drivers").asComponent().get("NiagaraNetwork");
106     niagaraNetwork.add("ProvisioningNwExt", new BProvisioningNiagaraNetworkExt());
107     BBatchJobService batchService = new BBatchJobService();
108     station.getServices().add("BatchJobService", batchService);
109
110     // Initialize another station as the provisioning target
111     targetStation = new BNiagaraStation();
112     targetStation.setAddress(BOrd.make("ip:localhost"));
113     BFoxClientConnection foxConnection = targetStation.getClientConnection();
114     foxConnection.setPort(foxPort);
115     foxConnection.setUseFoxs(false);
116     foxConnection.setCredentials(new BUsernameAndPassword("provisioning", "Password10"));
117
118     // Configure a test platform connection for the target station
119     // This test set does not utilize the platform connection, but it is included here as an example
120     BTestPlatformConnection connection = new BTestPlatformConnection();
121     BDaemonSession session = makeDaemonSession(false, false, true);
122     connection.setDaemonSession(session);
123     targetStation.add("provisioningNiagara_PlatformConnection", connection);
124
125     niagaraNetwork.add(station.getStationName(), targetStation);
126
127     // BHistoryJobSummaryManager creates the necessary histories at startup asynchronously.
128     // This shouldn't be an issue in normal circumstances, but in the tests where we run
129     // batch jobs immediately after startup, they may finish before the histories are created,
130     // causing a NullPointerException. Wait until this process completes.
131     for (int i = 0; i < 20 && !batchService.whenServiceStarted().isDone(); i++)
132     {
133         Thread.sleep(100);
134     }
135     if (!batchService.whenServiceStarted().isDone())
136     {
137         throw new SkipException("Timed out waiting for batch service to initialize.");
138     }

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-rt.gradle
 - module.lexicon
 - module-include.xml
 - module-permissions.xml
 - moduleTest-include.xml
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection
 - rc
 - app.ace
 - BasicAceApp.ace
 - catalog_test.json
 - MinimalAceApp.ace
 - SimpleAceApp.ace
 - gradle
 - wrapper
 - niagara.gradle
 - .gitattributes
 - .gitignore
 - build.gradle
 - environment.gradle

```

172  /**
173  * Reset the connection util Fox session before each test method
174  *
175  * @throws Exception
176  */
177  @BeforeMethod
178  public void beforeMethod() throws Exception
179  {
180      station = stationHandler.getStation();
181      try (ProvisioningConnectionUtil util = new ProvisioningConnectionUtil(targetStation, null))
182      {
183          util.getEngagedFoxSession().disconnect();
184      }
185  }
186
187
188  /**
189  * Run a job step and return the results
190  *
191  * @param step the job step to run
192  * @return the results
193  */
194  private BJobStepDetails runStep(BDeviceJobStep step)
195  {
196      BNiagaraNetworkJob job = new BNiagaraNetworkJob();
197      BSimple deviceId = BNetworkBatchAgent.get(getNiagaraNetwork(), null).getBatchDeviceId(targetStation);
198      job.setDevicesToProcess(BValueList.make(deviceId));
199      BForEachStationStage stage = job.getForEachStationStage();
200      stage.addStep(step);
201
202      BStation station = stationHandler.getStation();
203      station.add("job", job);
204      try
205      {
206          BBatchJobService batchService = (BBatchJobService) Sys.getService(BBatchJobService.TYPE);
207          BDeviceStepDetails details = step.run(batchService, targetStation, (DeviceNetworkJobOp) job.makeOp(null));
208          for (int i = 0; i < job.log().getItems().length; ++i)
209          {
210              LOGGER.info(" JobLogItem[" + i + "] = \"" + job.log().getItems()[i] + "\"");
211          }
212
213          return details;
214      }
215      finally
216      {
217          station.remove("job");
218      }
219  }
220

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection
 - rc
 - app.ace
 - BasicAceApp.ace
 - catalog_test.json
 - MinimalAceApp.ace
 - SimpleAceApp.ace

```

220 /**
221  * Make a mock daemon session that will respond to requests based on the specified parameters
222  *
223  * @param defaultCredentials true if the daemon session should return default credentials with auth servlet request
224  * @param defaultSysPw true if the daemon session should return default system passphrase with systempw requests
225  * @param stationRunning true if the daemon session should return a running station with applist requests
226  * @return the daemon session
227  * @throws IOException if an error occurs writing XML responses
228  */
229
230 @private static BDaemonSession makeDaemonSession(boolean defaultCredentials, boolean defaultSysPw, boolean stationRu
231 {
232     // The Mockito.spy() method will only mock specific methods we configure, otherwise the actual methods are called
233     BDaemonSession session = Mockito.spy(BDaemonSession.makeIgnoringCache(BLocalHost.INSTANCE, 3011));
234     BHostProperties hostProperties = session.getHostProperties();
235     hostProperties.setDaemonVersion("4.13");
236     hostProperties.setSslSupported(true);
237     hostProperties.getServletNames().add("systempw", BString.make("systempw"));
238     hostProperties.getServletNames().add("applist", BString.make("applist"));
239
240     // Configure platform message contents for specific platform daemon message types
241     ByteBuffer authBuffer = new ByteBuffer();
242     XWriter auth = new XWriter(authBuffer.getOutputStream());
243     auth.w("<authInfo>");
244     auth.w("<auth>");
245     auth.w("<user").w(' ').attr("name", "foo");
246     if (defaultCredentials)
247     {
248         auth.w(' ').attr("default", "true");
249     }
250     auth.w(">");
251     auth.w("</auth>");
252     auth.w("</authInfo>");
253     auth.flush();
254     auth.close();
255     byte[] authBytes = authBuffer.getBytes();
256
257     ByteBuffer syspwBuffer = new ByteBuffer();
258     XWriter syspw = new XWriter(syspwBuffer.getOutputStream());
259     syspw.w("<systemPassword");
260     if (defaultSysPw)
261     {
262         syspw.w(' ').attr("default", "true");
263     }
264     syspw.w(">");
265     syspw.flush();
266     syspw.close();
267     byte[] syspwBytes = syspwBuffer.getBytes();
268

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jslint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb

Project Structure

- summit413 [niagara]
 - aceAppProv
 - aceAppProv-rt [niagara.aceAppProv-rt]
 - src [main] sources root
 - com.acme.aceAppProv
 - BAceAppInstallStep
 - aceAppProv-rt.gradle
 - module.lexicon
 - module-include.xml
 - module-permissions.xml
 - moduleTest-include.xml
 - aceAppProv-wb [niagara.aceAppProv-wb]
 - src [main] sources root
 - com.acme.aceAppProv.ui
 - BAceAppInstallFactory
 - srcTest [niagaraModuleTest] sources
 - com.acme.aceAppProv.test
 - BAceAppInstallStepTest
 - BTestPlatformConnection
 - rc
 - app.ace
 - BasicAceApp.ace
 - catalog_test.json
 - MinimalAceApp.ace
 - SimpleAceApp.ace
 - aceAppProv-wb.gradle
 - module.lexicon
 - module-include.xml
 - module-permissions.xml
 - moduleTest-include.xml
- gradle
 - wrapper
 - niagara.gradle
 - .gitattributes
 - .gitignore
 - build.gradle
 - environment.gradle

```

309 @Test
310 public void testAceAppInstallToEmptyStation() throws Exception
311 {
312     BAceAppInstallStep step = new BAceAppInstallStep();
313     step.setAceFileName(aceToInstall.getFileName());
314     BJobStepDetails details = runStep(step);
315
316     String jobMessages = Arrays.asList(
317         details.log().getItems() List<JobLogItem>
318         .stream() Stream<JobLogItem>
319         .map( item -> { return item.getMessage(); } ) Stream<String>
320         .reduce("\n", (partialString, element) -> partialString + element + '\n');
321
322     Assert.assertEquals(details.getState(), BJobState.failed,
323         String.format("Job failed: %s", jobMessages));
324
325     Assert.assertTrue(jobMessages.contains("The necessary ACE device was not found on target device"));
326 }
327
328 @Test(dependsOnMethods = "testAceAppInstallToEmptyStation") // We want to run the empty station test first
329 public void testAceAppInstall() throws Exception
330 {
331     // Add ACE Network and Device to target station
332     BComponent stationRoot = targetStation.getComponentSpace().getRootComponent();
333     BComponent drivers = stationRoot.get("Drivers").asComponent();
334     drivers.add("AceEdgeNetwork", aceNetwork);
335     BAceDevice aceDevice = aceNetwork.getLocal();
336     aceDevice.add("App", new BAceApp());
337
338     // Set up some Mocking for unit testing the install step
339     BAceAppInstallStep step = Mockito.spy(new BAceAppInstallStep());
340     Mockito.when(step.getTargetObject(ArgumentMatchers.any())).thenReturn(aceDevice);
341     Mockito.doAnswer( invocation ->
342     {
343         BAceAppDownloadJob job = invocation.getArgument(0);
344         job.complete(BJobState.success);
345         return null;
346     }
347     ).when(step).waitForJobComplete(ArgumentMatchers.any(BAceAppDownloadJob.class));
348
349     // Run the step
350     step.setAceFileName(aceToInstall.getFileName());
351     BJobStepDetails details = runStep(step);
352
353     String jobMessages = Arrays.asList(
354         details.log().getItems() List<JobLogItem>
355         .stream() Stream<JobLogItem>
356         .map( item -> { return item.getMessage(); } ) Stream<String>
357         .reduce("\n", (partialString, element) -> partialString + element + '\n');

```

Gradle

- niagara
 - Tasks
 - build
 - build setup
 - documentation
 - help
 - niagara
 - migrateSlotomatic
 - moduleTestJar
 - slotomatic
 - other
 - verification
 - check
 - jshint
 - niagaraTest
 - test
 - Run Configurations
 - aceAppProv-rt
 - aceAppProv-wb



Additional Considerations

- Parallel job execution
 - getParallelExecutionConflicts()
- Provision device with default credentials
 - BIPrivilegedDeviceJobStep
- Security Manager
 - ProvisioningConnectionUtil
 - module-permissions.xml

```
@Override
public Set<BDevice> getParallelExecutionConflicts(BBatchJobService svc,
    BDevice device, Set<BDevice> allDevices, DeviceNetworkJobOp op)
{
    return Collections.singleton(device);
}
```

```
@NiagaraType
public class BRenameStationStep
    extends BDeviceJobStep
    implements BIPrivilegedDeviceJobStep
{
    /*+ ----- BEGIN BAJA AUTO GENERATED CODE ----- +*/
}
```

```
<niagara-permission-groups type="station">
  <req-permission> // Required permission
    <name>KEY_STORE</name>
    <purposeKey>This module needs to check exemptions.</purposeKey>
    <parameters>
      <parameter name="keystores" value="userExemptionStore"/>
      <parameter name="actions" value="read"/>
    </parameters>
  </req-permission>
</niagara-permission-groups>
```

github.com/tridium

The background of the entire image is a nighttime city skyline. The buildings are illuminated with various lights, creating a vibrant blue and white color palette. The sky is a deep, dark blue. The overall scene is a dense urban environment with many skyscrapers and smaller buildings.

NS2022

ACCELERATING INNOVATION