Summary

This page's goal is to help out with the steps needed to create a feature that contains a customer customization.
In short the process of creating a feature that contain a customization is pretty easy and involves

- Create a feature project
- Configure the feature project
- Build the feature

Create a feature

In Eclipse, use the function "New…" to create a "Feature project"	It is good to use a similar name scheme as the projects it should contain. Ex. The plugins • se.solme.avix.custom.volvo.common • se.solme.avix.custom.volvo.migration • se.solme.avix.custom.volvo.vps are packaged in the feature • se.solme.avix.custom.volvo.feature

Configuration

General information

There are some general information that needs to be setup for all features. Some of it are visible to users during the installation procedure while some are more related to administration and the technical stuff in the P2-framework and Eclipse.

Name	**Where?**	**Comment**
ID	"Overview" tab in feature.xml	Unique identifier. A tip is to use the project name. Ex. se.solme.avix.custom.volvo.feature
Version	"Overview" tab in feature.xml	Version of the feature. Increase each release. For "final" releases to customers, use at least "1.0.0.qualifier". During development before release a "0.X.X.qualifier" version is acceptable.
Name	"Overview" tab in feature.xml	Name of the feature. This is visible to end users so make it good and try to keep it short. Ex. "Volvo VPS Reports"
Provider	"Overview" tab in feature.xml	Set it to "Solme AB" as we are the source of the deliverable.
Feature Description	"Information" tab in feature.xml.	Description of what the feature does. This is visible to end users. Ex: Adaptations in the scope of VPS. These adaptations provide a

		new custom tab "Volvo AB" on tasks and some additional reports - "CL", "OIS", "SOP", "Stickers" and "TRI".
Copyright Notice	"Information" tab in feature.xml	Copyright text. This is visible to end users. If nothing special is said, use "Copyright Solme AB, all rights reserved."
License Agreement	"Information" tab in feature.xml	The EULA that end users must accept during install. If nothing special is required, steal it from the project "se.solme.avix.feature"

Included plugins/fragments/features

Add the plugins/fragments/features that should be part of the feature.

Where?	Comment
"Plug-ins" tab in feature.xml	Add the plugins and fragments that should be packaged in the feature. If there are some restrictions to the plugins set them up as well (normally not needed)
"Included Features" tab in feature.xml	Add the external features that should be packaged in the feature. Usually we leave this blank.

Dependencies

Setup requirements on the target environment to make sure that the feature is not installed in an AviX where it will not work.

Where?	**Comment**
"Dependencies" tab in feature.xml	Setup dependencies to plugins/features that must be present in the target before the feature can be installed. Most custom plugins should at least require an AviX v4.4.1 or later. Ex: se.solme.avix.feature | 4.4.1 Greater of Equal

Root files

Root files are files and directories apart from the included plugins/fragments/features that should be copied into the installation directory or into subfolders of the installation directory

Examples:

- BIRT reports
- Dlls
- AviX template files (.avx4t)

The folders and files that should be included as root files should be placed in the feature project. When 4.3-customizations are ported to 4.4 this means that custom BIRT-report folders need to be moved to the feature project.

Where?	**Comment**
build.properties	The file "build.properties" is accessible both as an own file and via the tab "build.properties" in the feature.xml-editor Folders and files are handled a little differently.

Copy folder structure and all files in it	root.folder.<**existing** destination folder in AviX> =<source folder in feature>
Copy single file	root.folder.<**existing** destination folder in AviX> =file:<source file in feature>

Example: The feature project "se.solme.avix.custom.volvo.feature" has some extra files and folders that should be included as root files when the feature is installed.

reports/Volvo VPS/changelist/*.*	Birt report. Should be installed in the folder "reports/Volvo VPS/changelist" in AviX.
reports/Volvo VPS/grouped_ois/*.*	Birt report. Should be installed in the folder "reports/Volvo VPS/grouped_ois" in AviX.
reports/Volvo VPS/standardoperationprocedure/*.*	Birt report. Should be installed in the folder "reports/Volvo VPS/standardoperationprocedure" in AviX.
reports/Volvo VPS/TRI/*.*	Birt report. Should be installed in the folder "reports/Volvo VPS/TRI" in AviX.
templates/Powertrain/AviX44PowertrainTemplate.avx4t	Custom template file. Should be installed in the folder "configuration/Powertrain" in AviX.
templates/VPS/AviX44VPSTemplate.avx4t	Custom template file. Should be installed in the folder "configuration/VPS" in AviX.

To get all birt reports in the folder "reports" in the feature to be installed in "reports" in AviX the following row is added to "build.properties"
root.folder.reports=reports

	To get all the template files in the folder "templates" in the feature installed in "configuration" in AviX the following row is added to "build.properties" root.folder.configuration=templates

The "build.properties" file looks like this for this example feature. The first row was automatically added when the feature project was created.	bin.includes = feature.xml
	root.folder.reports=reports
	root.folder.configuration=templates

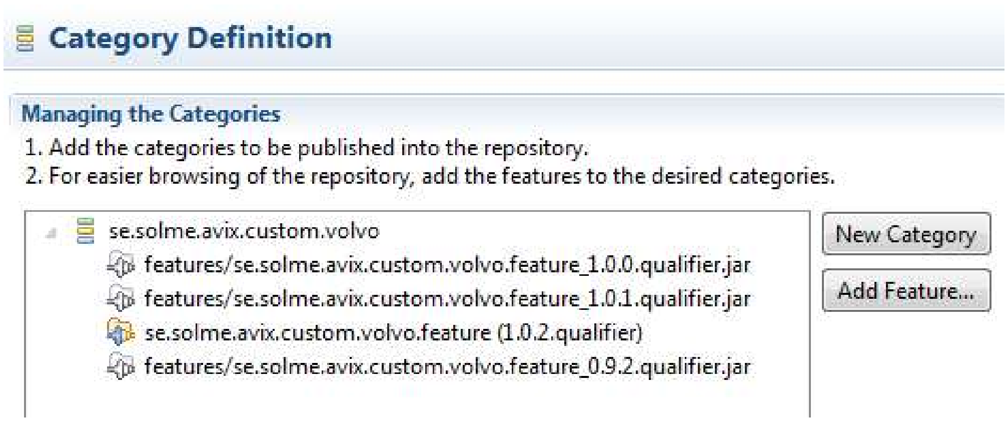
Build

Categories

Features can be organized in categories and we want our end user features to support that.
To support categories, add a category.xml file to the feature, for example by using "New..." in Eclipse and add a "Category Definition" to the feature project.

Edit the "category.xml" file and

- Create a category
- Give the category an ID, a Name and a Description
- Add your feature to the category. (The feature need to be added again each time the feature changes version and should be build)
  Example



Category Definition

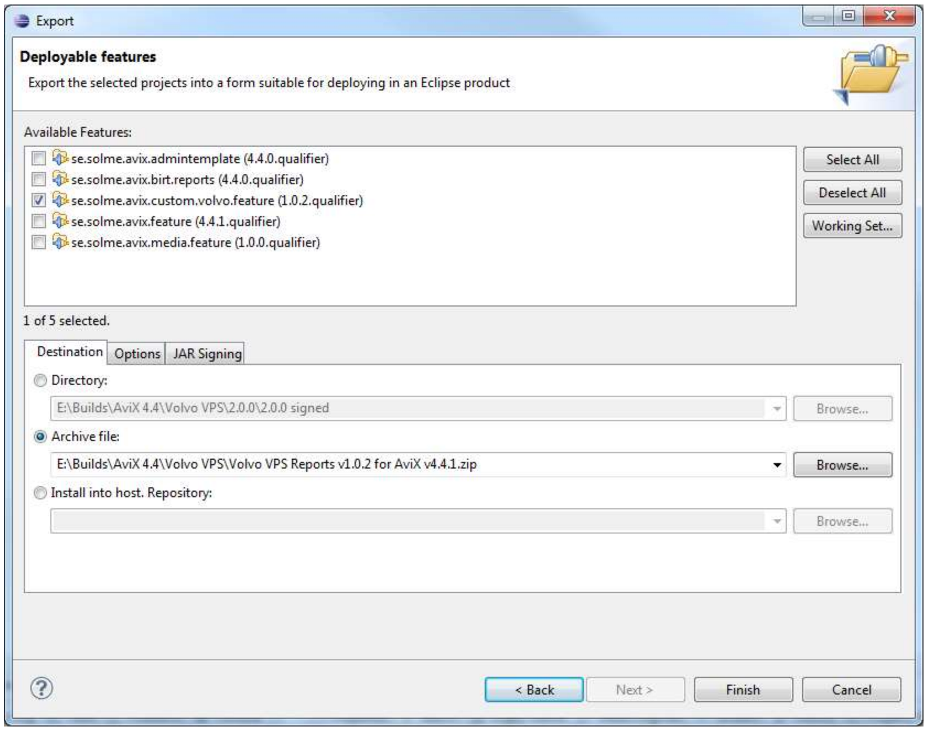
Managing the Categories
1. Add the categories to be published into the repository.
2. For easier browsing of the repository, add the features to the desired categories.

```
se.solme.avix.custom.volvo
    features/se.solme.avix.custom.volvo.feature_1.0.0.qualifier.jar
    features/se.solme.avix.custom.volvo.feature_1.0.1.qualifier.jar
    se.solme.avix.custom.volvo.feature (1.0.2.qualifier)
    features/se.solme.avix.custom.volvo.feature_0.9.2.qualifier.jar
```

Export

Build the feature into a deployable package in the form of a zip-file that can be used as an update site.

Right click the feature project and select "Export..."
Select the type "Deployable features"
In the tab "Destination" in the dialog, select "Archive file" and give the archive a name.


Export

Deployable features

Export the selected projects into a form suitable for deploying in an Eclipse product

Available Features:

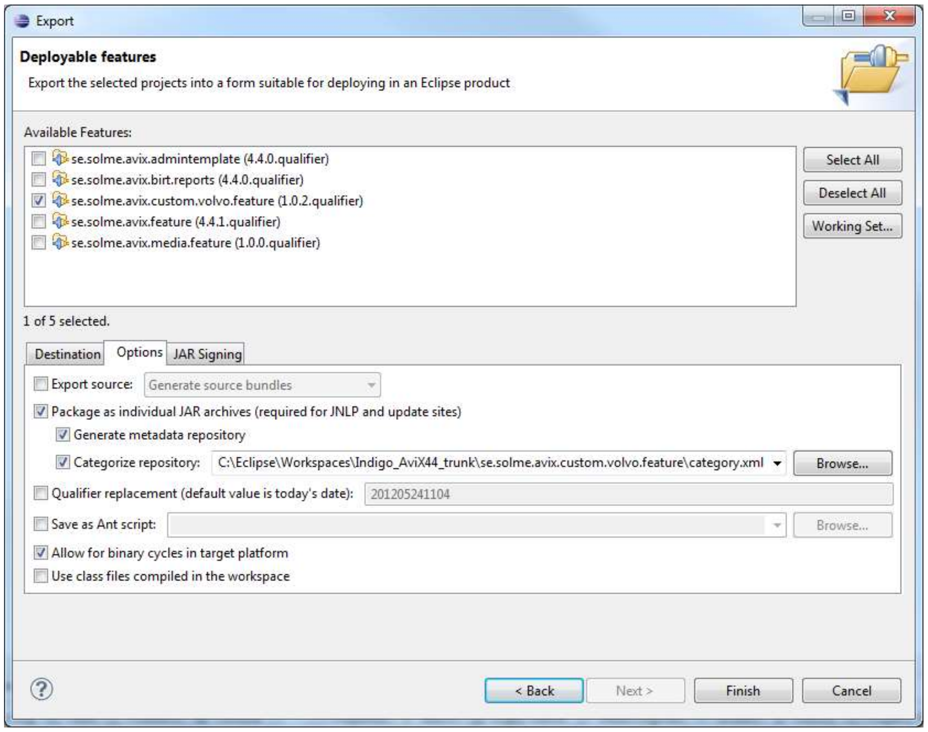
- [ ] se.solme.avix.admintemplate (4.4.0.qualifier)
- [ ] se.solme.avix.birt.reports (4.4.0.qualifier)
- [✓] se.solme.avix.custom.volvo.feature (1.0.2.qualifier)
- [ ] se.solme.avix.feature (4.4.1.qualifier)
- [ ] se.solme.avix.media.feature (1.0.0.qualifier)

1 of 5 selected.

Destination | Options | JAR Signing

- Directory: E:\Builds\AviX 4.4\Volvo VPS\2.0.0\2.0.0 signed
- [● Archive file:] E:\Builds\AviX 4.4\Volvo VPS\Volvo VPS Reports v1.0.2 for AviX v4.4.1.zip
- Install into host. Repository:

In the tab "Options" in the dialog, make sure "Generate metadata repository" and "Categorize repository" is active. Assign your "category.xml" file as value for "Categorize repository".


Export

Deployable features

Export the selected projects into a form suitable for deploying in an Eclipse product

Available Features:

- [ ] se.solme.avix.admintemplate (4.4.0.qualifier)
- [ ] se.solme.avix.birt.reports (4.4.0.qualifier)
- [✓] se.solme.avix.custom.volvo.feature (1.0.2.qualifier)
- [ ] se.solme.avix.feature (4.4.1.qualifier)
- [ ] se.solme.avix.media.feature (1.0.0.qualifier)

1 of 5 selected.

Destination | Options | JAR Signing

- [ ] Export source: Generate source bundles
- [✓] Package as individual JAR archives (required for JNLP and update sites)
  - [✓] Generate metadata repository
  - [✓] Categorize repository: C:\Eclipse\Workspaces\Indigo_AviX44_trunk\se.solme.avix.custom.volvo.feature\category.xml
- [ ] Qualifier replacement (default value is today's date): 201205241104
- [ ] Save as Ant script:
- [✓] Allow for binary cycles in target platform
- [ ] Use class files compiled in the workspace

If the jars in the feature should be signed, open the tab "JAR Signing". We dont have a real certificate at the moment so we have not used signed jars in the past, but that may change in the future. The benefit of having signed jars (and signed with a trusted certificate) is that users do not get an extra dialog when they install/update components.

If no signing should be done, make sure that the checkbox "Sign the JAR archives using a keystore" is not checked.

Signing can only be done when you run Eclipse in a JDK so you need to start it with a flag to force that when you want to sign jars.
Example:
"C:\Eclipse\Indigo\eclipse.exe" -vm "C:\Program Files (x86)\Java\jdk1.6.0_26\bin\javaw.exe"

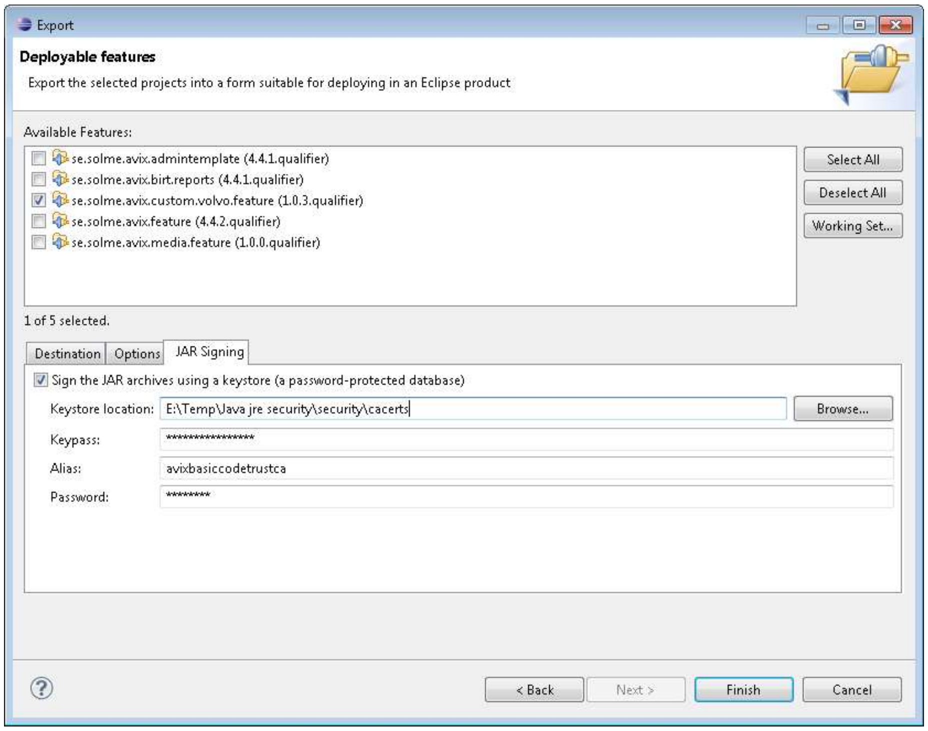
The signing parameters are:

	Description	Latest certficate (valid to Oct 2020)	Parameters that use our Comodo code signing certificate
Keystore location	The keystore that should be used	Z:\Utveckling\Certificates\SolmeKeyStore	Z:\Utveckling\Certificates\SolmeKeyStore
Keypass	The password set on the alias when it was created	Y7rrsJF0pMSPLzunGyKn	knNZaAI8q2qEZUwkSvt5
Alias	The name of the alias in the keystore that should be used for the signing	te-bde7f813-44e9-414d-bbaa-dc96fde6c45f	solme ab's comodo ca limited id
Password	The password for the keystore	knNZaAI8q2qEZUwkSvt5	knNZaAI8q2qEZUwkSvt5

The old parameters below are still preferred to use for AviX 4.6 releases, because the new ones cause issues with the old JRE!

	Description	Parameters that works with our shipped JRE from AviX v4.4.2

Keystore location	The keystore that should be used	Z:\Utveckling\AviX4\Build\jre\lib\security\cacerts
Keypass	The password set on the alias when it was created	AviX4rockarfett!
Alias	The name of the alias in the keystore that should be used for the signing	avixbasiccodetrustca
Password	The password for the keystore The default cacert-file in java has the password "changeit"	changeit



Export

Deployable features

Export the selected projects into a form suitable for deploying in an Eclipse product

Available Features:

- [ ] se.solme.avix.admintemplate (4.4.1.qualifier)
- [ ] se.solme.avix.birt.reports (4.4.1.qualifier)
- [x] se.solme.avix.custom.volvo.feature (1.0.3.qualifier)
- [ ] se.solme.avix.feature (4.4.2.qualifier)
- [ ] se.solme.avix.media.feature (1.0.0.qualifier)

1 of 5 selected.

Destination | Options | JAR Signing

- [x] Sign the JAR archives using a keystore (a password-protected database)

Keystore location: E:\Temp\Java jre security\security\cacerts

Keypass: ****************

Alias: avixbasiccodetrustca

Password: ********

Note:
As we ship AviX bundled with the JRE that AviX uses we can in theory ship that JRE with a cacert-file that contain a self signed certificate and by that enable us to use that self signed certificate to sign JARs in our projects without users getting the "Do you trust this self signed certificate" dialog. This may cause problems if we want to update the JRE though. Need to investigate it further.

Press "Finish" to export the feature.