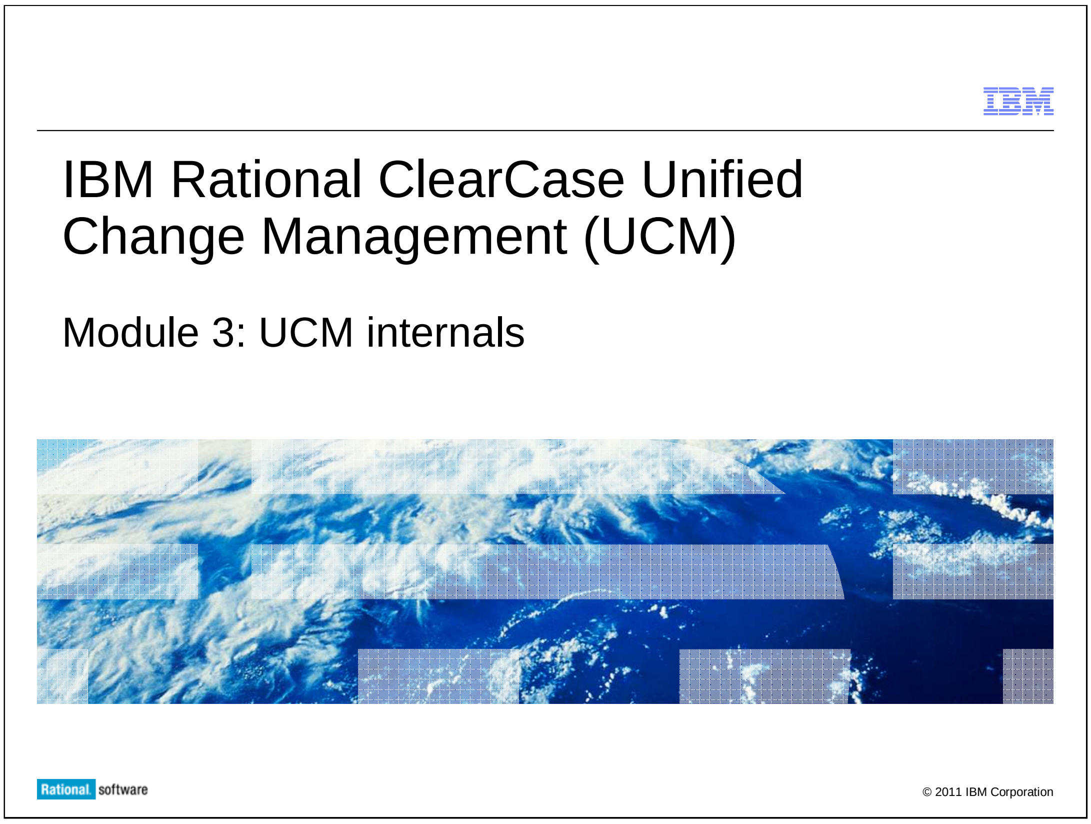
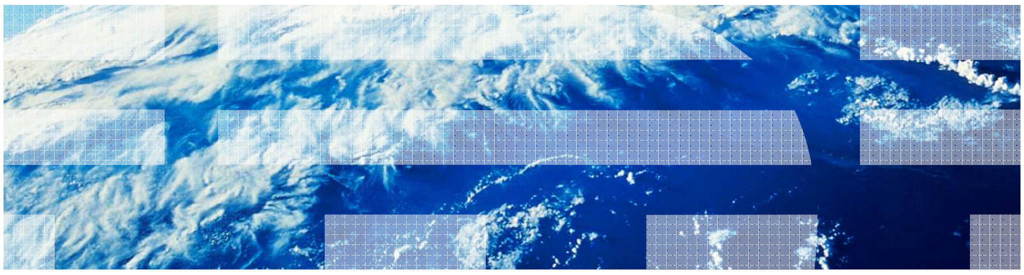


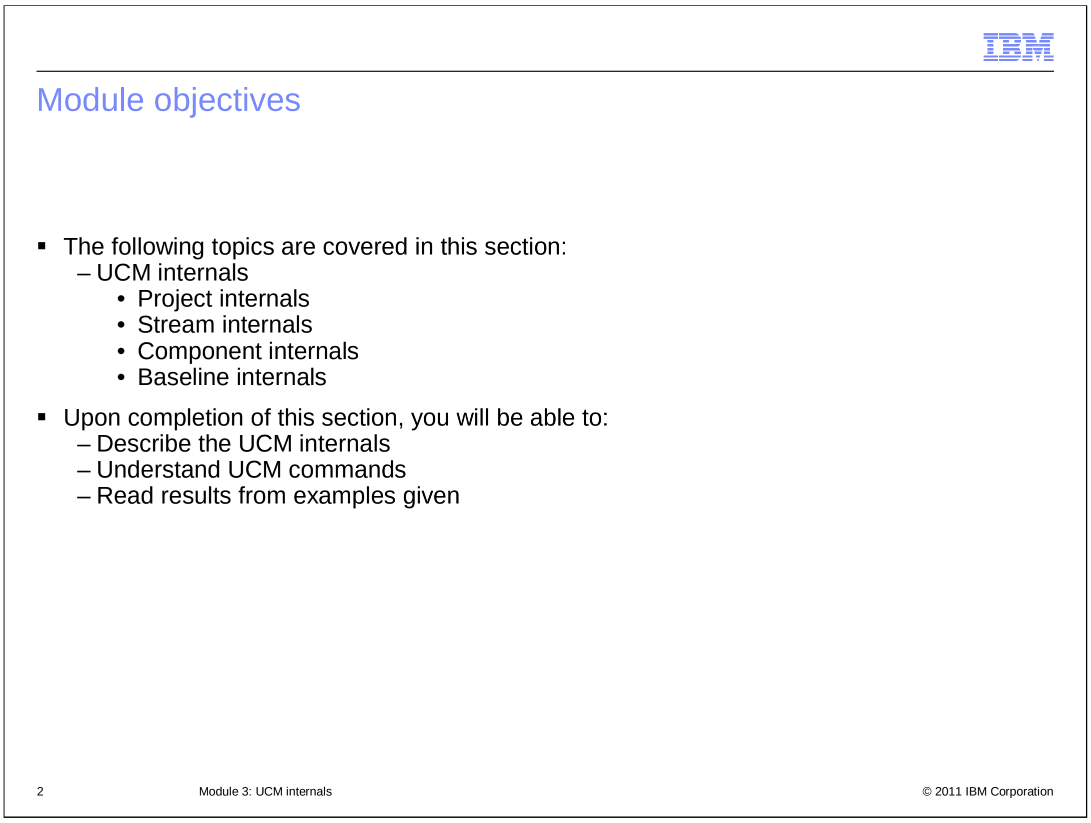
IBM Rational ClearCase Unified Change Management (UCM)

Module 3: UCM internals



Rational. software

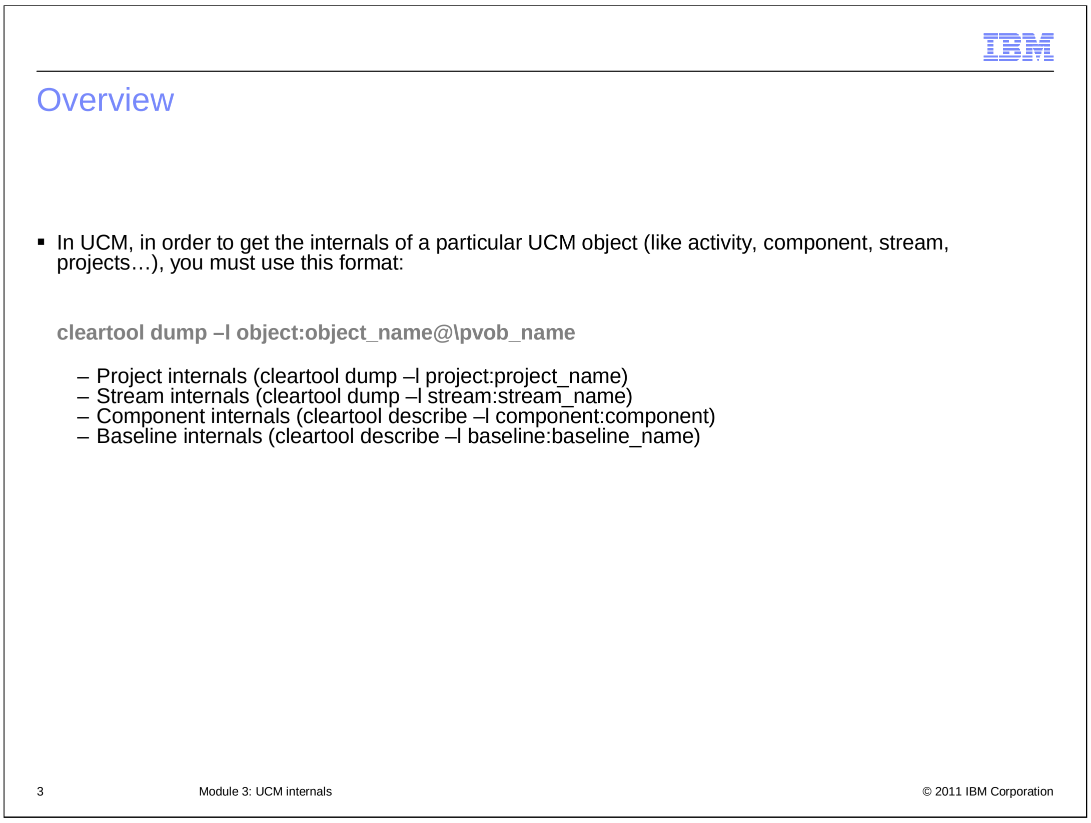
This presentation looks at UCM internals.



Module objectives

- The following topics are covered in this section:
  - UCM internals
    - Project internals
    - Stream internals
    - Component internals
    - Baseline internals
- Upon completion of this section, you will be able to:
  - Describe the UCM internals
  - Understand UCM commands
  - Read results from examples given

This module covers UCM internals, including: project internals, stream internals, component internals, and baseline internals. Upon completion of this section, you will be able to describe the UCM internals, understand UCM commands, and read results from examples given in this module.



Overview

- In UCM, in order to get the internals of a particular UCM object (like activity, component, stream, projects…), you must use this format:

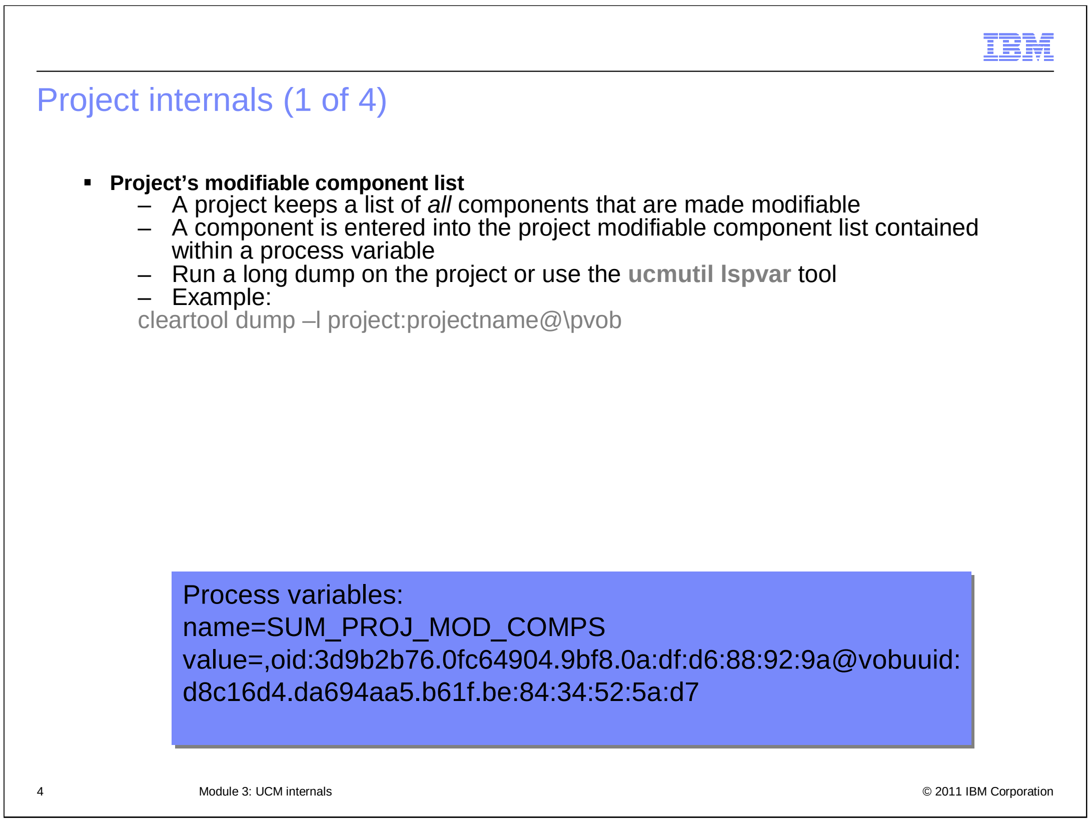
  cleartool dump –l object:object_name@\pvob_name

  – Project internals (cleartool dump –l project:project_name)
  – Stream internals (cleartool dump –l stream:stream_name)
  – Component internals (cleartool describe –l component:component)
  – Baseline internals (cleartool describe –l baseline:baseline_name)

3 Module 3: UCM internals © 2011 IBM Corporation

In UCM, you can get internals from any UCM object; for example, an activity, a component, a stream, a project, and so on. In UCM, in order to get the internals of a particular UCM object, you must follow the format shown here.

These examples give you an idea of how to get internals information from project, stream, component, and baseline internals, including specifics like the modifiable component list or the configuration specifications covered in the next few slides.



IBM

Project internals (1 of 4)

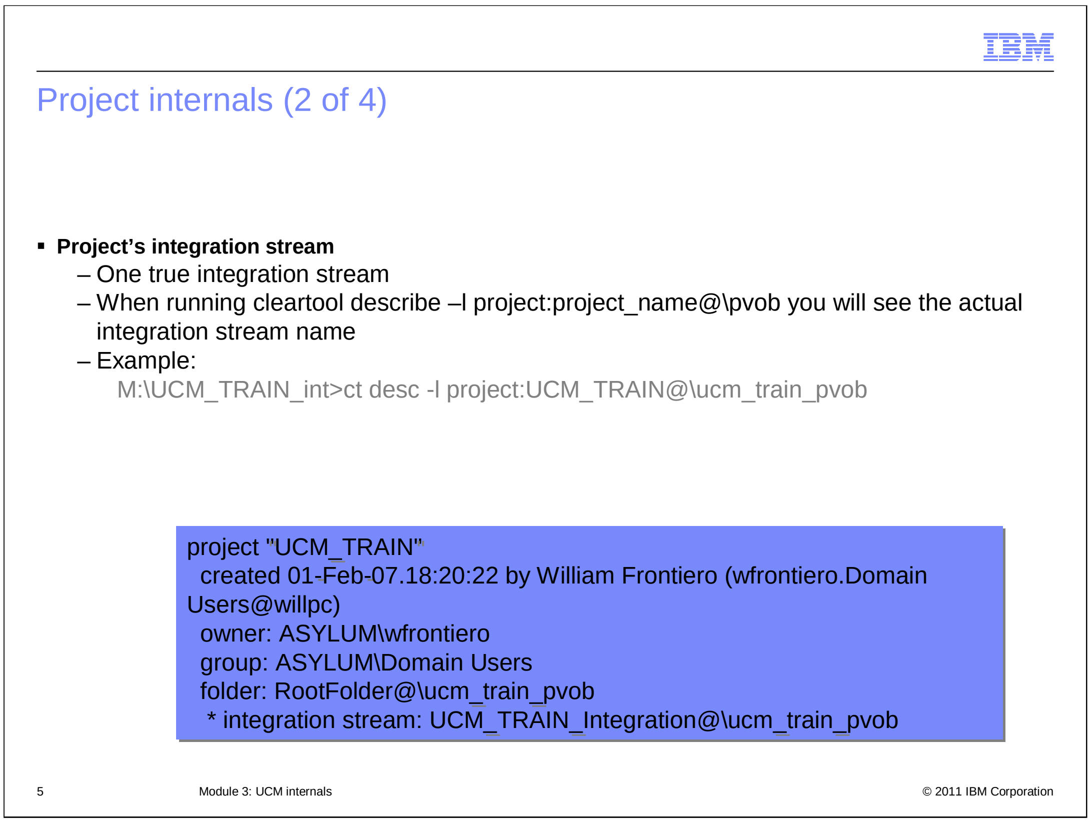
- **Project's modifiable component list**
  - A project keeps a list of *all* components that are made modifiable
  - A component is entered into the project modifiable component list contained within a process variable
  - Run a long dump on the project or use the **ucmutil lspvar** tool
  - Example:

  cleartool dump –l project:projectname@\pvob

```
Process variables:
name=SUM_PROJ_MOD_COMPS
value=,oid:3d9b2b76.0fc64904.9bf8.0a:df:d6:88:92:9a@vobuuid:
d8c16d4.da694aa5.b61f.be:84:34:52:5a:d7
```

A project keeps a list of *all* components that are made modifiable. Once a component is made modifiable it can not be changed back and is very difficult to remove from the project. The component is entered into the project's *modifiable component list* contained within a process variable. This process variable can be listed by running a long dump on the project or using the ucmutil Lspvar tool.

This example shows running the cleartool dump command. In the results, look for the "Process variables:" section of the output. In this example you can see the oid of the component "@" the vobuuid of the pvob. It can be translated to value = ucmtraining@\ucm_train_pvob. You can run a "cleartool describe dash L" on the long string to see which component the value is referring to.



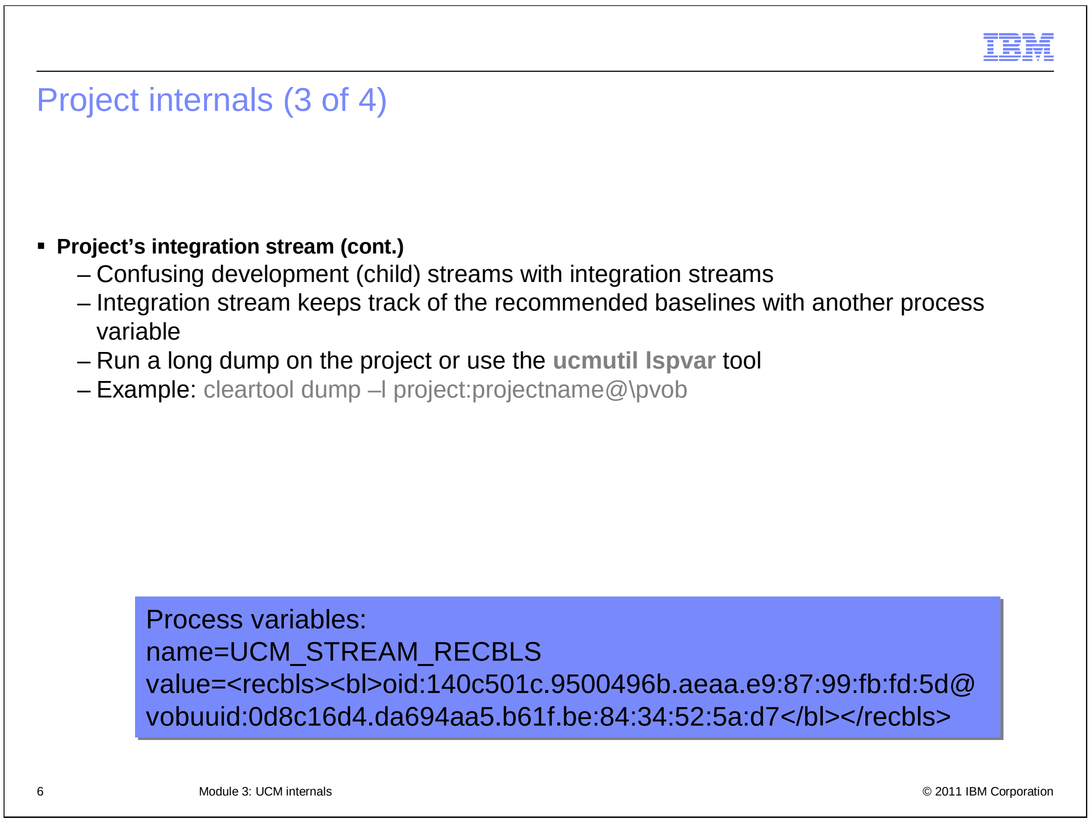
Project internals (2 of 4)

- **Project's integration stream**
  - One true integration stream
  - When running cleartool describe –l project:project_name@\pvob you will see the actual integration stream name
  - Example:

    M:\UCM_TRAIN_int>ct desc -l project:UCM_TRAIN@\ucm_train_pvob

```
project "UCM_TRAIN"
  created 01-Feb-07.18:20:22 by William Frontiero (wfrontiero.Domain
Users@willpc)
  owner: ASYLUM\wfrontiero
  group: ASYLUM\Domain Users
  folder: RootFolder@\ucm_train_pvob
   * integration stream: UCM_TRAIN_Integration@\ucm_train_pvob
```

While you can use development stream as an integration, every project has only ONE true integration stream. You can see what the actual integration stream is by running a **cleartool describe –l**ong on the project. The results are shown here. You can see the integration stream on the last line.



Project internals (3 of 4)

- **Project's integration stream (cont.)**
  - Confusing development (child) streams with integration streams
  - Integration stream keeps track of the recommended baselines with another process variable
  - Run a long dump on the project or use the **ucmutil lspvar** tool
  - Example: cleartool dump –l project:projectname@\pvob

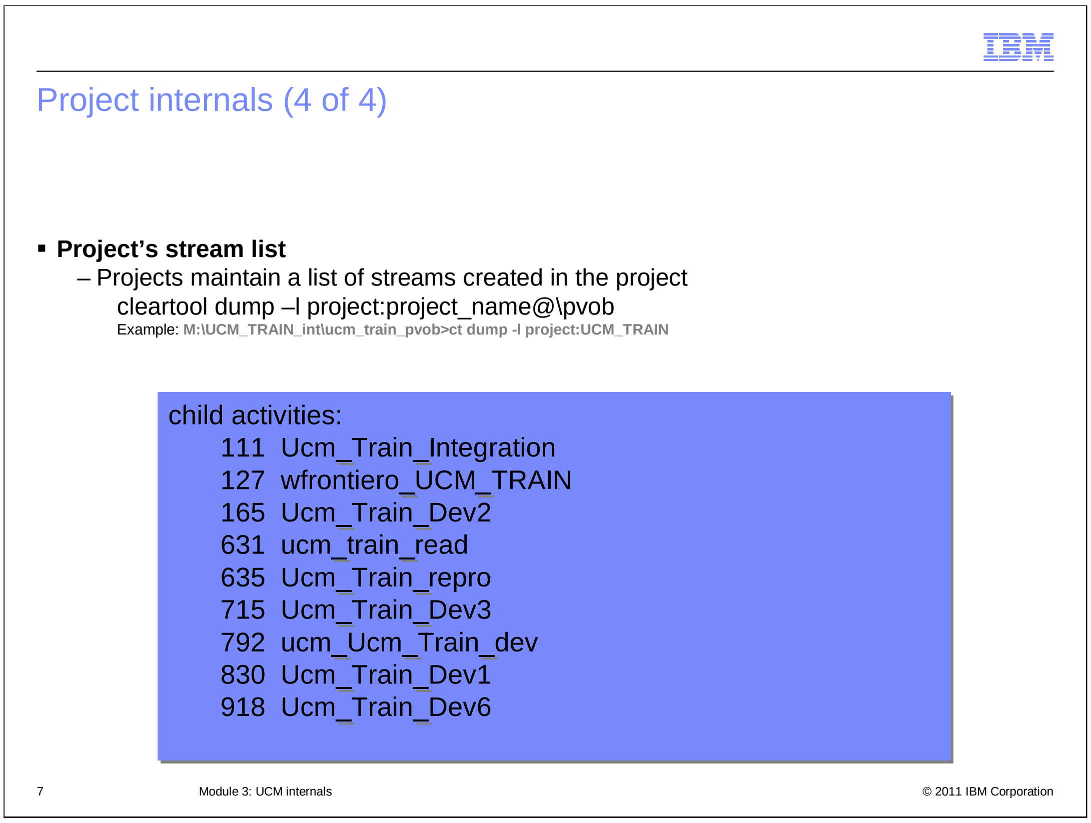
```
Process variables:
name=UCM_STREAM_RECBLS
value=<recbls><bl>oid:140c501c.9500496b.aeaa.e9:87:99:fb:fd:5d@
vobuuid:0d8c16d4.da694aa5.b61f.be:84:34:52:5a:d7</bl></recbls>
```

6 Module 3: UCM internals © 2011 IBM Corporation

Be sure not to confuse development (or, child) streams with integration streams. They are not the same. By default, the integration stream will keep track of the recommended baselines with another process variable. This process variable can be listed by running a long dump on the project or using the ucmutil Lspvar tool.

Note again that you can run "cleartool describe –l" on the long oid that is returned to see what baseline is being referenced.

In the results of the command shown here you will see each recommended baseline represented by an oid.



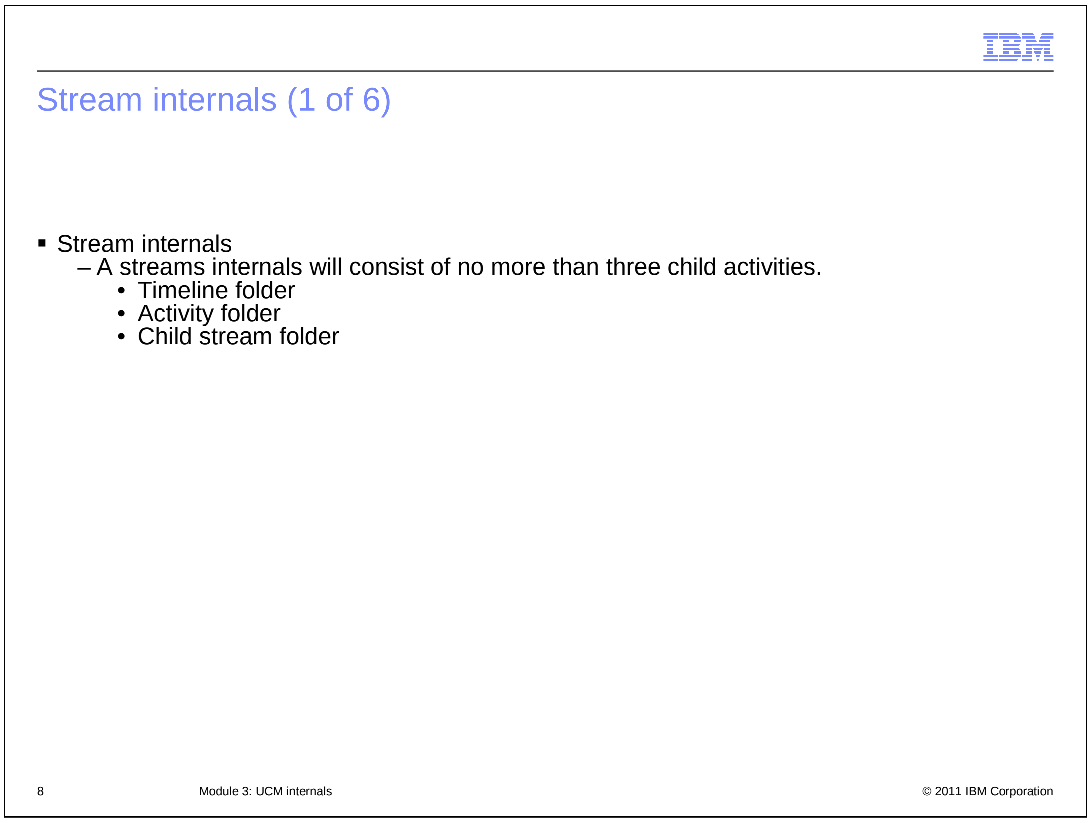
Project internals (4 of 4)

- **Project's stream list**
  – Projects maintain a list of streams created in the project
    cleartool dump –l project:project_name@\pvob
    Example: **M:\UCM_TRAIN_int\ucm_train_pvob>ct dump -l project:UCM_TRAIN**

```
child activities:
      111  Ucm_Train_Integration
      127  wfrontiero_UCM_TRAIN
      165  Ucm_Train_Dev2
      631  ucm_train_read
      635  Ucm_Train_repro
      715  Ucm_Train_Dev3
      792  ucm_Ucm_Train_dev
      830  Ucm_Train_Dev1
      918  Ucm_Train_Dev6
```

UCM projects maintain a list of streams created in the project. Use the cleartool dump command shown here to list the oids of the streams in the project.

The example command returned the UCM TRAIN results shown here. In the results you will see a number of child activities. Each child activity represents a steam in the project.

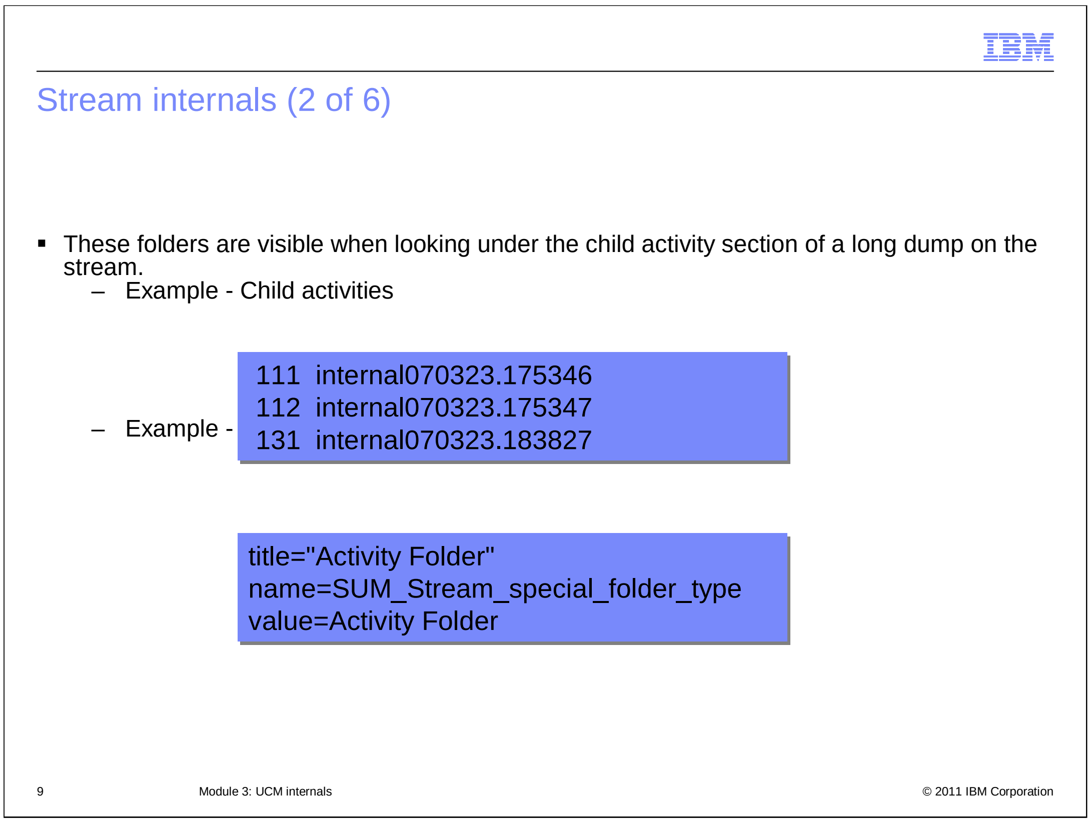


Stream internals (1 of 6)

- Stream internals
  - A streams internals will consist of no more than three child activities.
    - Timeline folder
    - Activity folder
    - Child stream folder

8 Module 3: UCM internals © 2011 IBM Corporation

Now that you have some exposure to project internals, this presentation will switch gears and look at stream internals. A stream’s internals will consist of no more than three child activities: the timeline, activity, and child stream folders. The timeline folder is keeps track of all deliveries, rebases and baselines on the stream. The timeline folder can be used to determine what components the stream has historically worked on. The activity folder keeps track of all activities that are associated with that stream, and the child stream folder keeps track of any child stream associated with that stream.



Stream internals (2 of 6)

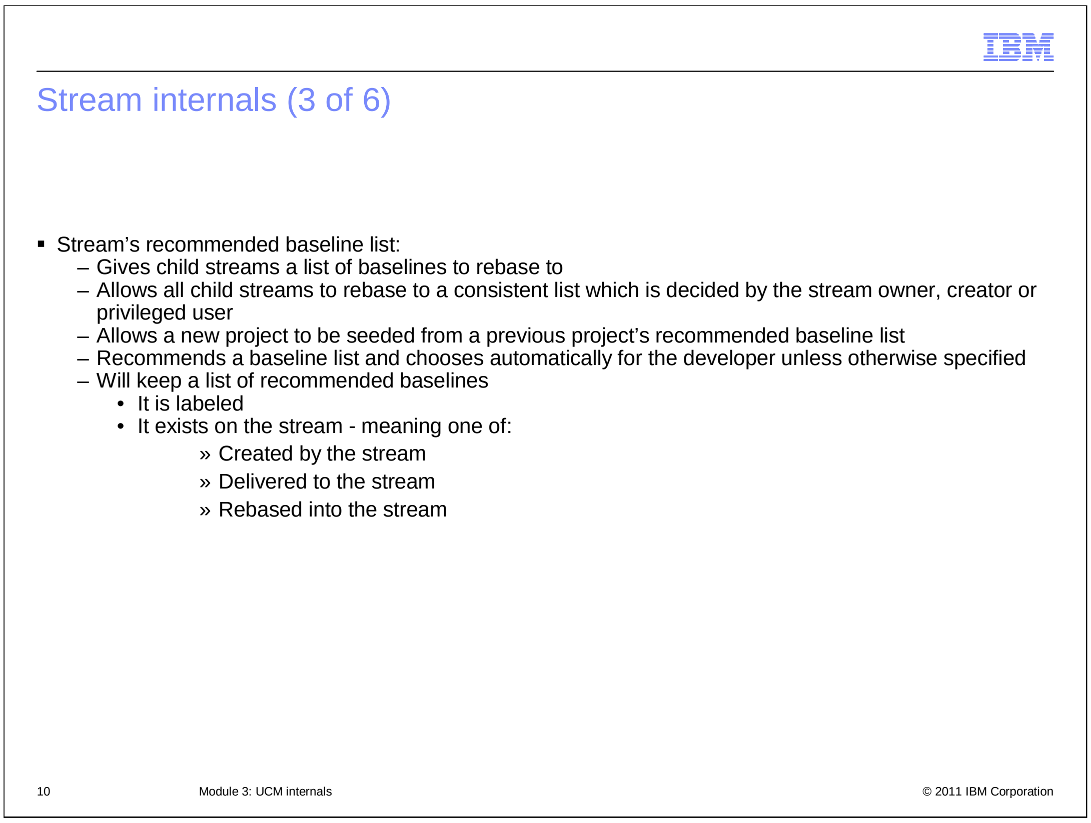
- These folders are visible when looking under the child activity section of a long dump on the stream.
  - Example - Child activities
  - Example -

```
111  internal070323.175346
112  internal070323.175347
131  internal070323.183827
```

```
title="Activity Folder"
name=SUM_Stream_special_folder_type
value=Activity Folder
```

These folders are visible when looking under the “child activity” section of a long dump on the stream. For example, when looking at the child activities, you will see the folders 111, 112, and 131. In this example each line represents an internal folder of the stream. You can determine which folder each represents by dumping the d-bid listed.

Other things to note: a checkpoint represents a make baseline, a foundation represents a rebasing of the stream (or, re-foundation), and an integration represents integration activity (or, deliver activity).



IBM

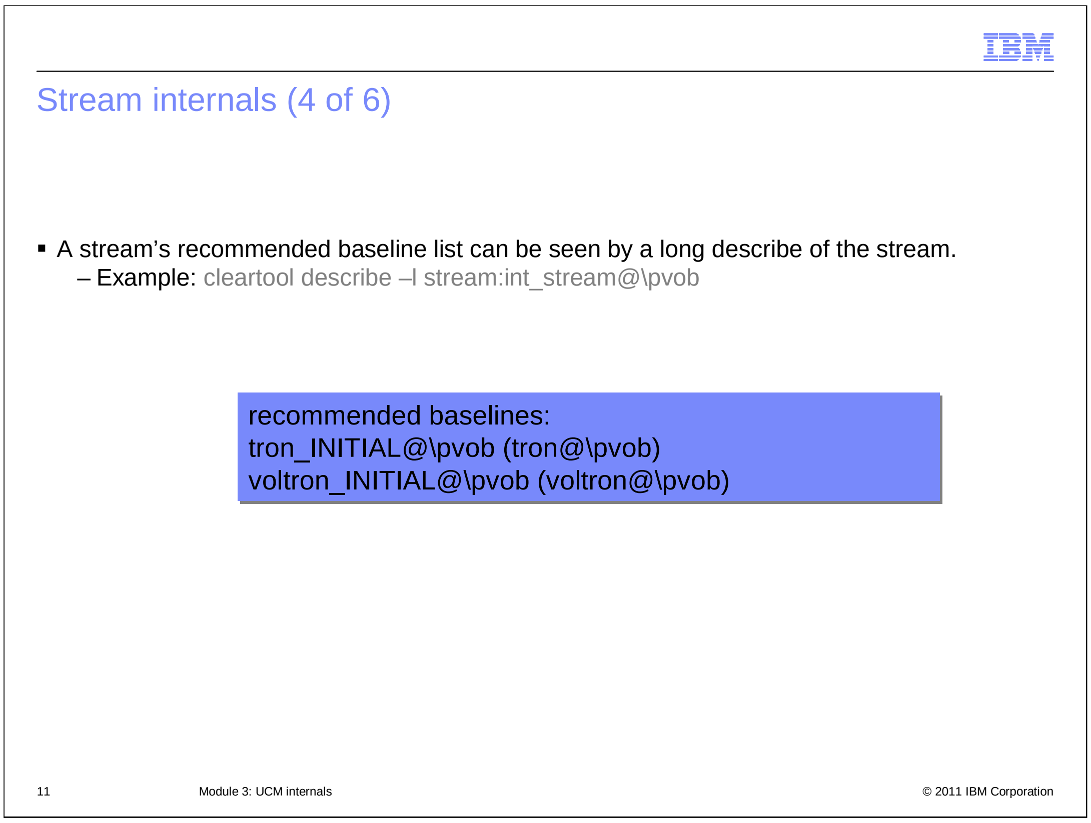
Stream internals (3 of 6)

- Stream’s recommended baseline list:
  - Gives child streams a list of baselines to rebase to
  - Allows all child streams to rebase to a consistent list which is decided by the stream owner, creator or privileged user
  - Allows a new project to be seeded from a previous project’s recommended baseline list
  - Recommends a baseline list and chooses automatically for the developer unless otherwise specified
  - Will keep a list of recommended baselines
    - It is labeled
    - It exists on the stream - meaning one of:
      - » Created by the stream
      - » Delivered to the stream
      - » Rebased into the stream

10 Module 3: UCM internals © 2011 IBM Corporation

A stream’s recommended baseline list is very helpful. A stream’s recommended baseline gives child streams a list of baselines to rebase to. It allows all child streams to rebase to a consistent list which is decided by the stream owner, creator or privileged user. It allows a new project to be seeded from a previous projects recommended baseline list. When a child stream attempts to rebase, the parent streams recommended baseline list is chosen automatically for the developer unless otherwise specified. And a stream will keep a list of recommended baselines.

Note that a baseline can be recommended if, and only if: 1) it is labeled, or, 2) it exists on the stream - meaning either that it is created by the stream, delivered to the stream, or rebased into the stream.



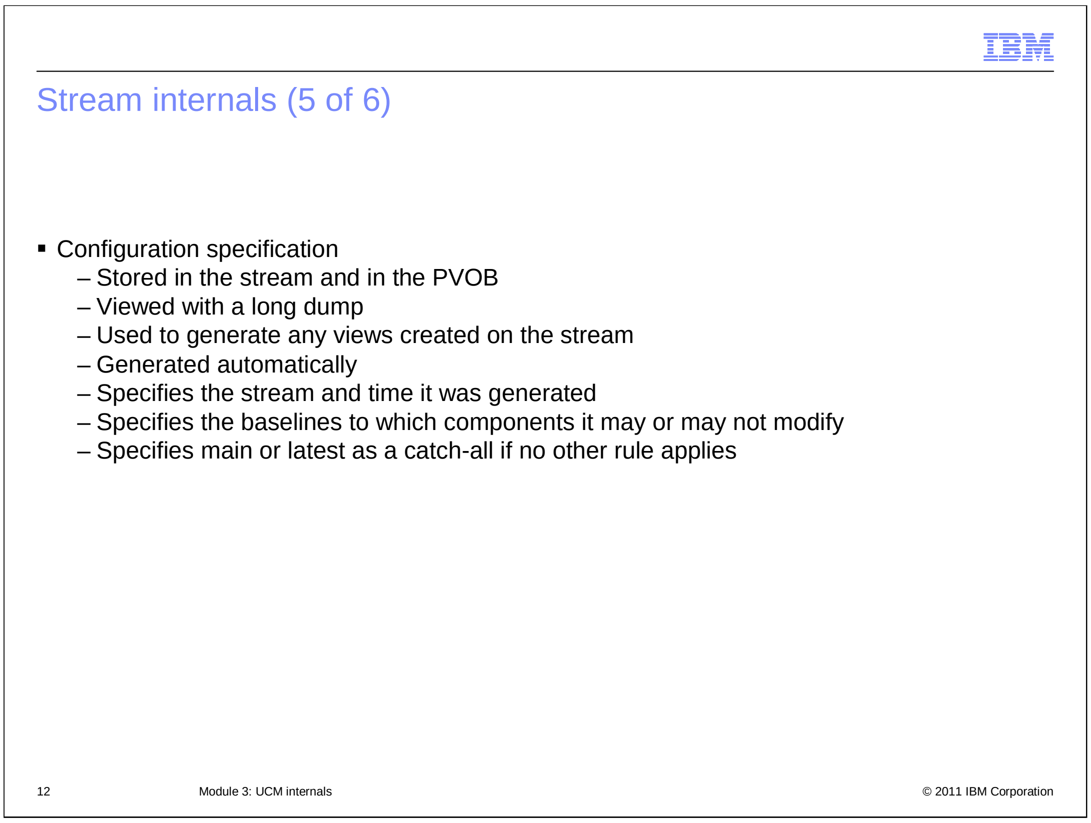
Stream internals (4 of 6)

- A stream’s recommended baseline list can be seen by a long describe of the stream.
  - – Example: cleartool describe –l stream:int_stream@\pvob

```
recommended baselines:
tron_INITIAL@\pvob (tron@\pvob)
voltron_INITIAL@\pvob (voltron@\pvob)
```

A stream’s recommended baseline list can be seen by a long describe of the stream. In this example, the describe command gives these results.

Look for the recommended baselines section similar to the one shown here for tron and voltron.



Stream internals (5 of 6)

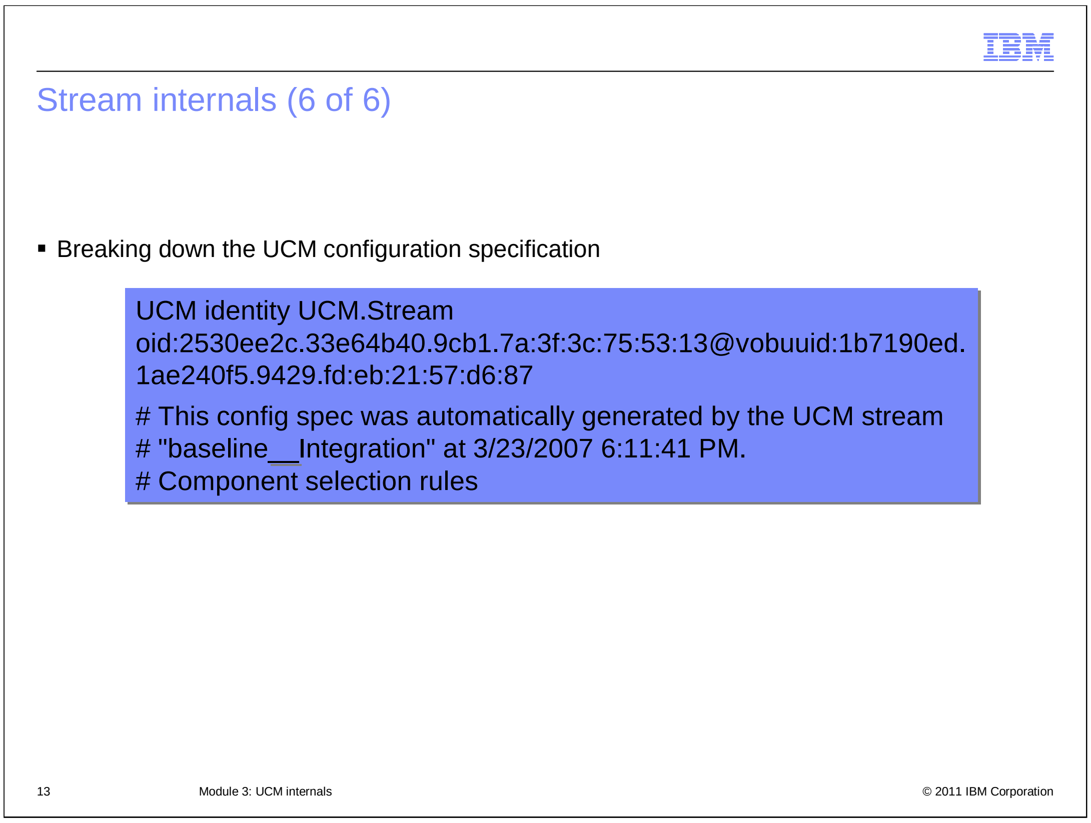
- Configuration specification
  - Stored in the stream and in the PVOB
  - Viewed with a long dump
  - Used to generate any views created on the stream
  - Generated automatically
  - Specifies the stream and time it was generated
  - Specifies the baselines to which components it may or may not modify
  - Specifies main or latest as a catch-all if no other rule applies

12 Module 3: UCM internals © 2011 IBM Corporation

A stream's *configuration specification* is stored in the stream and in the PVOB. You can view the configuration specification stored on the stream with a long dump; it is used to generate any views created on the stream.

When creating a view from a stream you will notice the view's configuration specification is identical to the dump dash long of the stream.

A UCM configuration specification is generated automatically, and remember that it is specific to stream. It specifies the stream and time it was generated, the baselines to which components it may or may not modify, and the main or latest as a catch-all if no other rule applies. Note that the UCM configuration specification will match that of the stream's configuration. Finally, note the view's configuration specification is identical to the dump –long of the stream.



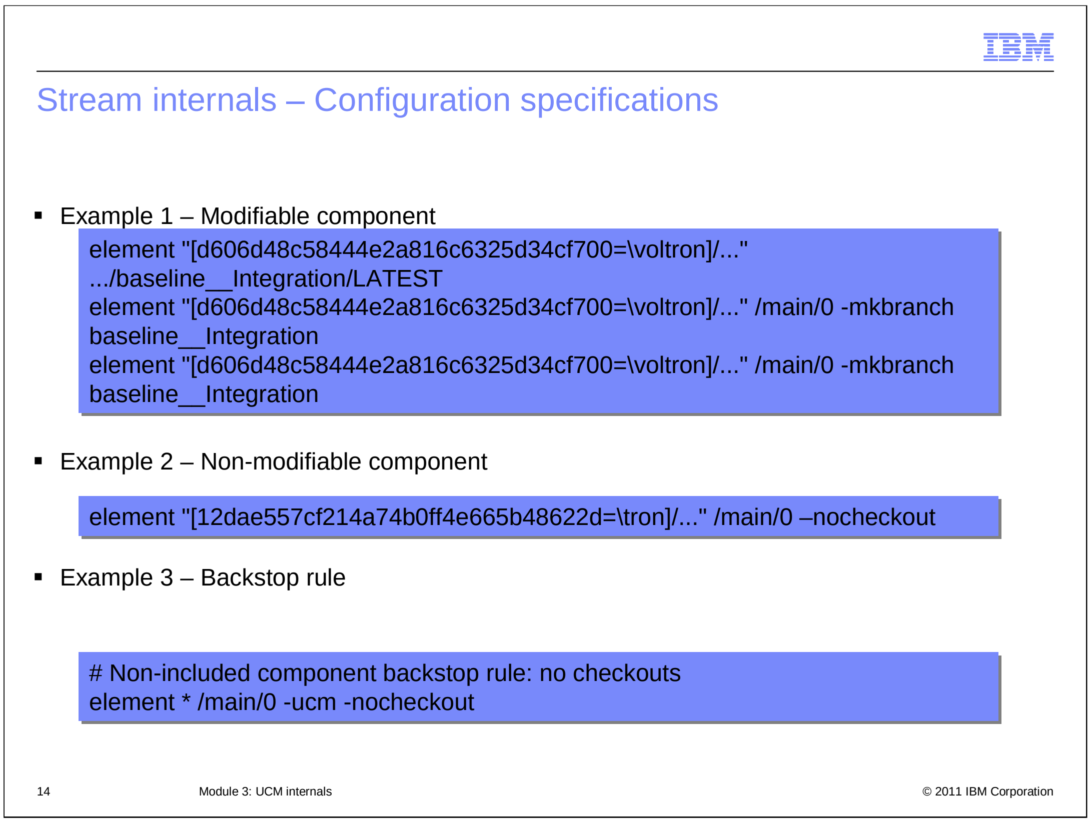
Stream internals (6 of 6)

- Breaking down the UCM configuration specification

```
UCM identity UCM.Stream
oid:2530ee2c.33e64b40.9cb1.7a:3f:3c:75:53:13@vobuuid:1b7190ed.
1ae240f5.9429.fd:eb:21:57:d6:87
# This config spec was automatically generated by the UCM stream
# "baseline__Integration" at 3/23/2007 6:11:41 PM.
# Component selection rules
```

13 Module 3: UCM internals © 2011 IBM Corporation

This example can tell you a lot of information you need to know about the UCM configuration specification. The first few lines identify the stream from which the configuration specification was generated. Then it tells you that the configuration specification was automatically generated by the UCM stream. Next, the stream name and last date the configuration specification was generated, and last, any component selection rules that apply. Those rules provide two key bits of information: 1) Which baseline the view believes it is looking at, and 2) If the component is modifiable in the stream.



Stream internals – Configuration specifications

- Example 1 – Modifiable component

```
element "[d606d48c58444e2a816c6325d34cf700=\voltron]/..."
.../baseline__Integration/LATEST
element "[d606d48c58444e2a816c6325d34cf700=\voltron]/..." /main/0 -mkbranch
baseline__Integration
element "[d606d48c58444e2a816c6325d34cf700=\voltron]/..." /main/0 -mkbranch
baseline__Integration
```

- Example 2 – Non-modifiable component

```
element "[12dae557cf214a74b0ff4e665b48622d=\tron]/..." /main/0 –nocheckout
```

- Example 3 – Backstop rule

```
# Non-included component backstop rule: no checkouts
element * /main/0 -ucm -nocheckout
```

14 Module 3: UCM internals © 2011 IBM Corporation

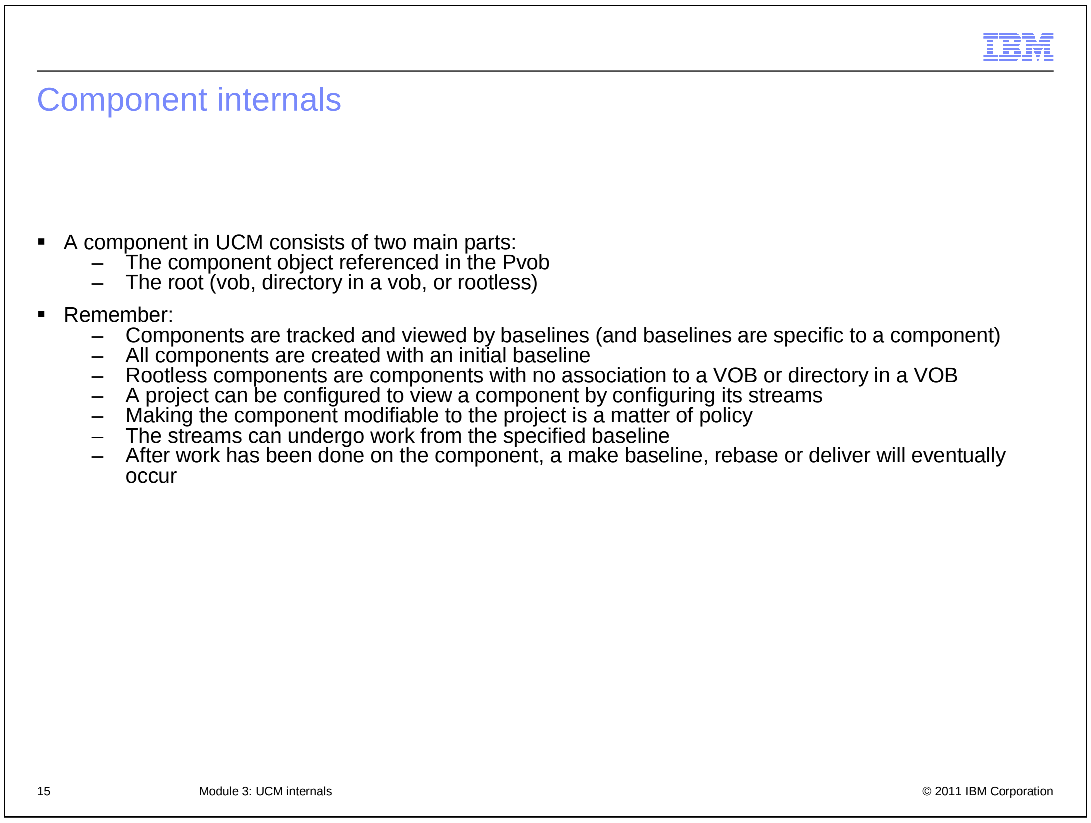
The first example shows what a *modifiable* component looks in the configuration specification.

Notice the three element selections for one element. This shows that the vob root “\voltron” is modifiable in this view.

The second example shows what a *non-modifiable* component looks like in the configuration specification.

Notice there is only one element selection rule for “\tron”. This shows that the vob root “\tron” is non-modifiable from this view.

In the last example, you will notice the backstop rule in the configuration specification. This indicates that no UCM checkouts can occur on the main branch.



Component internals

- A component in UCM consists of two main parts:
  - The component object referenced in the Pvob
  - The root (vob, directory in a vob, or rootless)
- Remember:
  - Components are tracked and viewed by baselines (and baselines are specific to a component)
  - All components are created with an initial baseline
  - Rootless components are components with no association to a VOB or directory in a VOB
  - A project can be configured to view a component by configuring its streams
  - Making the component modifiable to the project is a matter of policy
  - The streams can undergo work from the specified baseline
  - After work has been done on the component, a make baseline, rebase or deliver will eventually occur

15 Module 3: UCM internals © 2011 IBM Corporation

Now that you have some exposure to both project and stream internals, this presentation will switch gears and look at component internals.

Remember, a component in UCM consists of two main parts: The component object referenced in the PVOB and the root (vob, directory in a vob, or rootless). A component is tracked and viewed by baselines, and baselines are specific to a component.

All components are created with an initial baseline - this baseline can be used by any project within the specified PVOB.

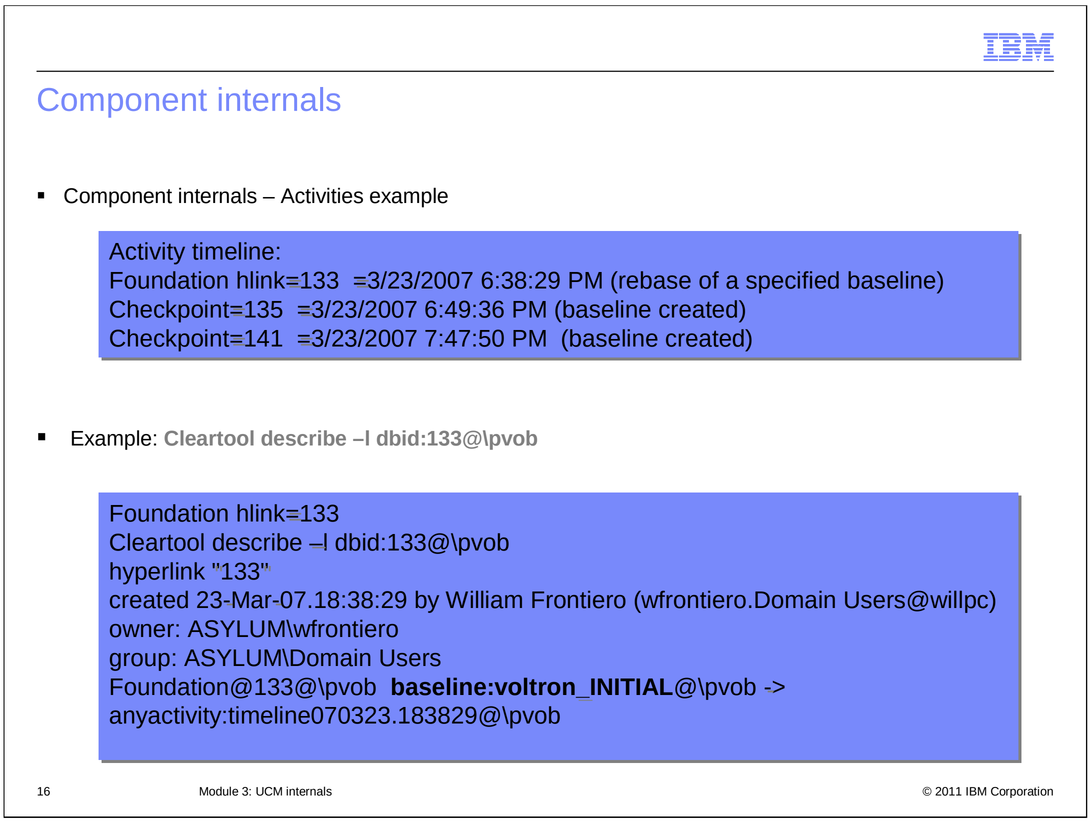
Rootless components are components with no association to a vob or directory in a vob. This means rootless components well never be modifiable.

A project can be configured to view a component by configuring its streams. The streams within the project would select a baseline of the component by means of a rebase.

Making the component modifiable to the project is a matter of policy. A component can be modifiable in one project and non-modifiable in another. Each project can decide if the component should be modifiable or not.

Once a component is made modifiable in a project, the streams can then undergo work from the specified baseline.

And last, after work has been done on the component, a make baseline, rebase or deliver will eventually occur. The moment one of these occur, a timeline instance for the component is created on the stream.



Component internals

- Component internals – Activities example

```
Activity timeline:
Foundation hlink=133  =3/23/2007 6:38:29 PM (rebase of a specified baseline)
Checkpoint=135  =3/23/2007 6:49:36 PM (baseline created)
Checkpoint=141  =3/23/2007 7:47:50 PM  (baseline created)
```

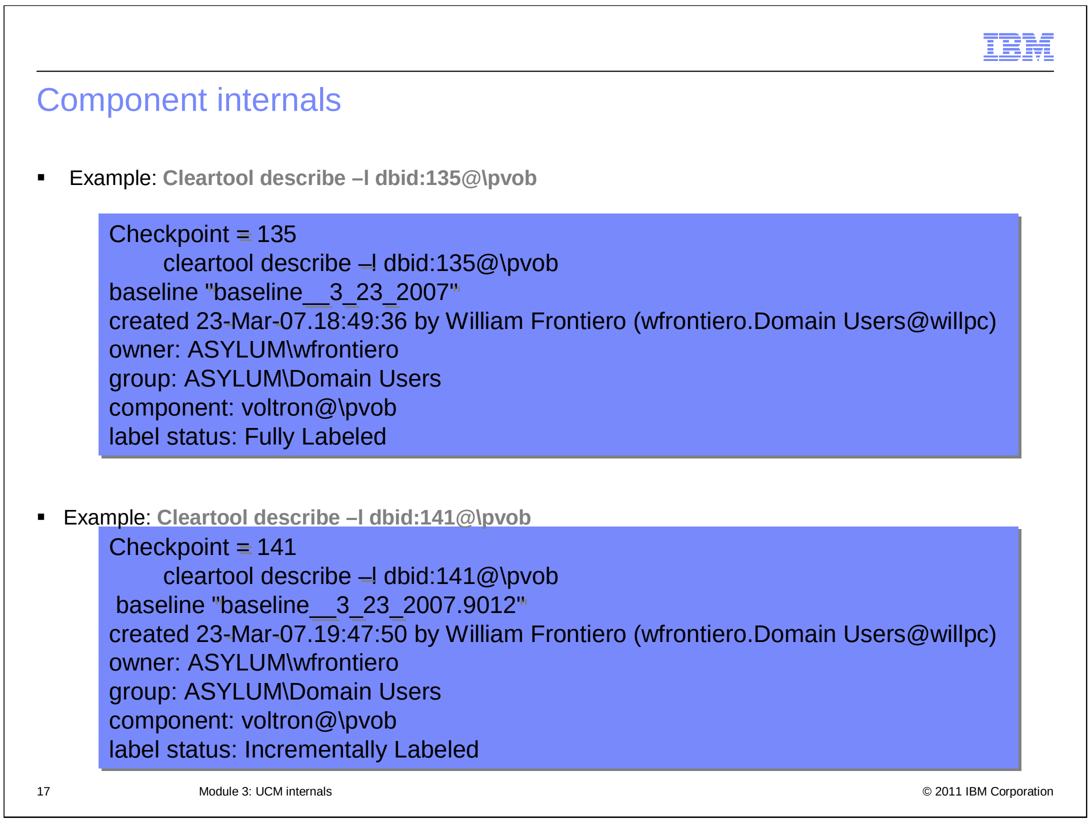
- Example: **Cleartool describe –l dbid:133@\pvob**

```
Foundation hlink=133
Cleartool describe –l dbid:133@\pvob
hyperlink "133"
created 23-Mar-07.18:38:29 by William Frontiero (wfrontiero.Domain Users@willpc)
owner: ASYLUM\wfrontiero
group: ASYLUM\Domain Users
Foundation@133@\pvob  baseline:voltron_INITIAL@\pvob ->
anyactivity:timeline070323.183829@\pvob
```

16 Module 3: UCM internals © 2011 IBM Corporation

In this example, recall that streams keep a timeline for each modifiable component on the stream. The timeline for this component will now log any new baselines, delivers or rebases on the stream and will list the history of this component's modification on the stream as shown here. Note the checkpoints, activities, and time stamps.

If you run a describe command, each instance listed in the example can be shown with the corresponding baselines or actions that occurred.



Component internals

- Example: **Cleartool describe –l dbid:135@\pvob**

```
Checkpoint = 135
        cleartool describe –l dbid:135@\pvob
baseline "baseline__3_23_2007"
created 23-Mar-07.18:49:36 by William Frontiero (wfrontiero.Domain Users@willpc)
owner: ASYLUM\wfrontiero
group: ASYLUM\Domain Users
component: voltron@\pvob
label status: Fully Labeled
```

- Example: **Cleartool describe –l dbid:141@\pvob**

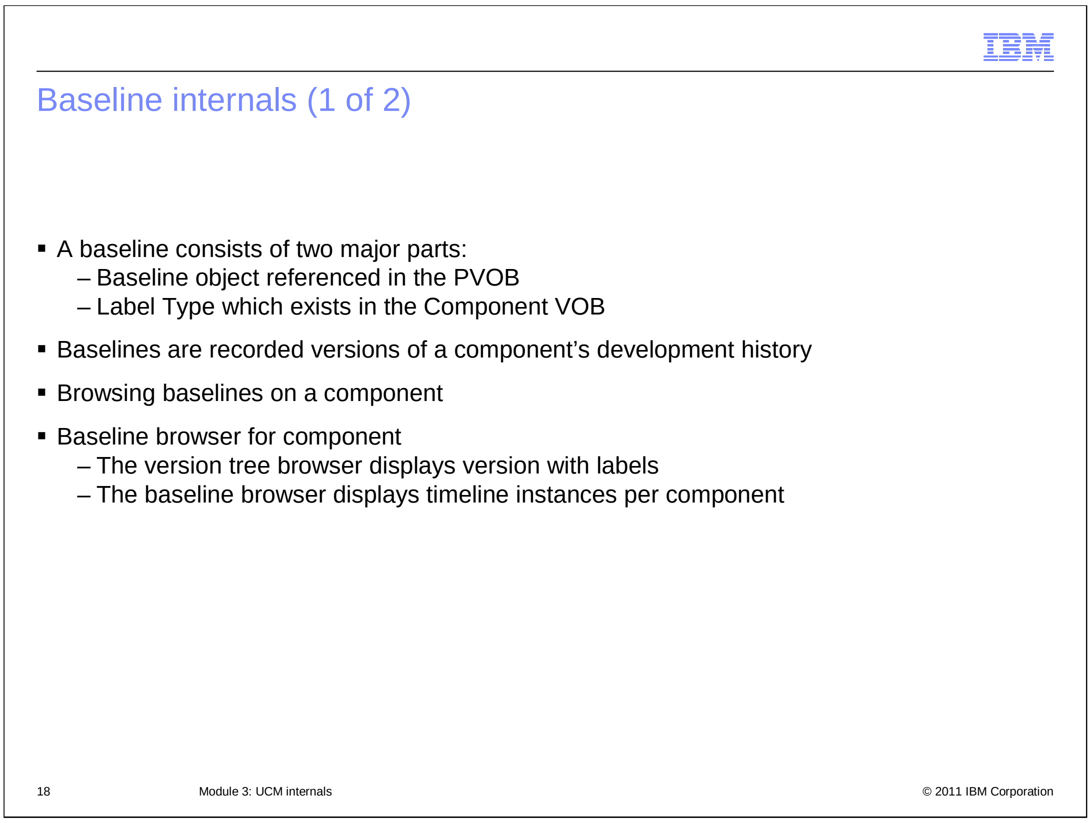
```
Checkpoint = 141
        cleartool describe –l dbid:141@\pvob
 baseline "baseline__3_23_2007.9012"
created 23-Mar-07.19:47:50 by William Frontiero (wfrontiero.Domain Users@willpc)
owner: ASYLUM\wfrontiero
group: ASYLUM\Domain Users
component: voltron@\pvob
label status: Incrementally Labeled
```

17 Module 3: UCM internals © 2011 IBM Corporation

Continuing from the last example, running describe on the remaining two checkpoints will yield these results.

[Pause 10 seconds]

The output here gives you information about the baseline.



Baseline internals (1 of 2)

- A baseline consists of two major parts:
  - Baseline object referenced in the PVOB
  - Label Type which exists in the Component VOB
- Baselines are recorded versions of a component’s development history
- Browsing baselines on a component
- Baseline browser for component
  - The version tree browser displays version with labels
  - The baseline browser displays timeline instances per component

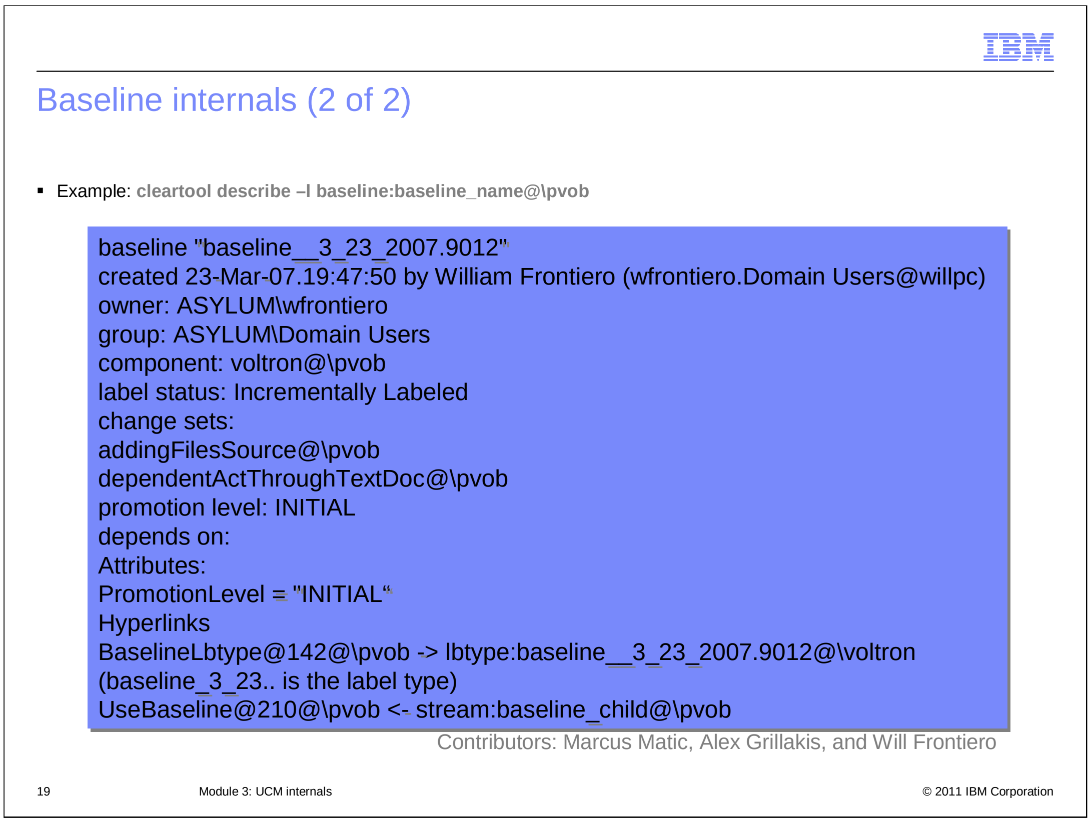
18 Module 3: UCM internals © 2011 IBM Corporation

Last on the internals discussion are baseline internals. A baseline consists of two major parts: 1) the baseline object referenced in the PVOB, and 2) the label type which exists in the component VOB.

The baseline object points to the label type which exists in the component root vob, and the label type is what is used to label versions created when developing in UCM. The label type is also used in the selection rules of UCM configuration specifications.

Baselines are nothing more than recorded versions of a component’s development history. When browsing baselines on a component you are viewing all the timelines of the component -- this is why you see the different streams which have created baselines and modified the component.

Finally, the baseline browser for a component is like that of a version tree of an element. The version tree browser displays versions with labels, while the baseline browser displays timeline instances per component.



Baseline internals (2 of 2)

- Example: **cleartool describe –l baseline:baseline_name@\pvob**

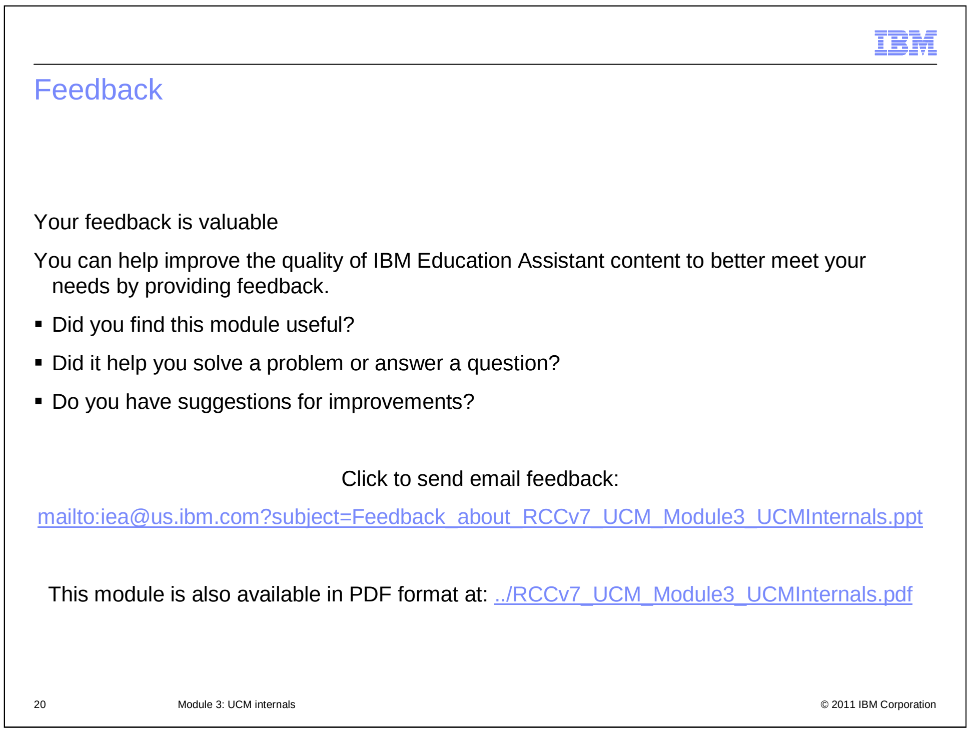
```
baseline "baseline__3_23_2007.9012"
created 23-Mar-07.19:47:50 by William Frontiero (wfrontiero.Domain Users@willpc)
owner: ASYLUM\wfrontiero
group: ASYLUM\Domain Users
component: voltron@\pvob
label status: Incrementally Labeled
change sets:
addingFilesSource@\pvob
dependentActThroughTextDoc@\pvob
promotion level: INITIAL
depends on:
Attributes:
PromotionLevel = "INITIAL"
Hyperlinks
BaselineLbtype@142@\pvob -> lbtype:baseline__3_23_2007.9012@\voltron
(baseline_3_23.. is the label type)
UseBaseline@210@\pvob <- stream:baseline_child@\pvob
```

Contributors: Marcus Matic, Alex Grillakis, and Will Frontiero

19 Module 3: UCM internals © 2011 IBM Corporation

The “component:” line shows this baseline is associated with the component voltron. The “label status:” line shows the baseline is currently incrementally labeled, and the “change sets” section shows the activities that are associated with the baseline.

The “Hyperlinks” section of the output displays what label type a baseline is associated with, and what streams are currently using the baseline. The “baselineLBtype” line shows that the label type is 3_23, and the “UseBaseline” line shows the baseline_child is currently looking at this particular baseline.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_RCCv7_UCM_Module3_UCMInternals.ppt

This module is also available in PDF format at: ../RCCv7_UCM_Module3_UCMInternals.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information