Job Control Story Report: JC8 – Define Job Criteria and Actions

Revision	Reason	Author	Date
0.01	Initial document	Peter Villiers	04Feb2011
0.02	Added information about the "not" condition	Ann Bagley	05Aug2011
0.03	Added managedServer to the process columns	Ann Bagley	15Nov2012

Story Summary:

As a Job Control Manager, I want to be able to define selection criteria for jobs and course of action to be taken on those jobs, so that jobs can be identified/high priority projects can be allowed more resources.

Preconditions:

- The Installer has installed the Job Controller client application.
- The server side of the application has been installed.
- The client side application has been tested to confirm that it works.
- The Job Control Manager has the ability to copy & edit the configuration files associated with the client application.

Story Path (Main):

The client side application has been installed and is known to work. The job control manager wants to define limits on jobs running in the system and the subsequent actions that are to be taken against the jobs that match the criteria. The user may also want to specify "globally exempt" jobs which are not subject to any of the criteria.

In the following text, JC_HOME is meant to be the location where the Job Controller client is installed.

There are 2 options for how to maintain the criteria and actions:

- The JC_HOME/conf/sddActions.xml file can be edited directly. This is the simplest in terms of initial setup. HOWEVER, if a new version of the client is installed, the file should be backed up prior to the installation. The tagging definitions will then need to be transferred to the newly installed file.
- The JC_HOME/conf/sddActions.xml file can be copied to another location and edited there. In this case, the JVM option "-Djc_action_defs=*the-path-to-the-custom-sddActions.xml*" can be added to the startup command to point to the custom file. This is the preferred option as it separates the installed application from the custom configuration.

The second option will be illustrated in the steps below. An example configuration directory will be shown however this could be anywhere:

1. Navigate to the parent directory of JC_HOME.
2. Create a directory called "JobController-customConfig".
3. Copy the sddActions.xml file from "JC_HOME/conf" to the "JobController-customConfig".
4. Edit the "JobController-config/sddActions.xml".
5. Save the file.
6. Remember to add the startup option to any scheduled jobs that should use the new tagging definitions.

Setting the Global Exclusions:

1. Edit the contents of the element SddJobController/Actions/GlobalExcludes to contain <ActionCondition> and/or <ActionBracket> elements. Multiple elements under the <GlobalExcludes> element are evaluated using an "OR" operator.

Adding a new Action Definition

The element SddJobController/Actions contains a sub-element for each Action.

1. Either:
    a. Copy/paste an existing element that is similar to what is needed, or
    b. Add a new one from scratch.

The Action logic evaluates the process using the <GlobalExclude>, <Exclude> and <Include> elements (in that order) to determine if a process meets the criteria and if so will mark the process with the <ActionConsequence> information and the <DescriptionText> value for execution and reporting.

In the following "@xxxxx" is meant to denote an XML attribute.

The main element <Action> can have the following sub-elements:

- <Description> - provides a location to hang
    - <DescriptionText> - A description of the condition set that is being tested for. Will be seen in emails. The separate text element allows for expansion to handle multiple languages later if need be.
- <ActionConditions> - defines the conditions that will be checked for the parent <Action> element.
    - <Include> - a set of <ActionCondition> and <ActionBraket> elements that are evaluated using an "OR" operator to determine if the process meets the criteria for inclusion.
        - <ActionCondition> - provides a way to test the information collected for the process.
            - @field – the name of the field to be tested. These are the same as the comment field in job history log file.
            - @comparator – provides the condition that will be used for the evaluation. The following table describes which conditions may be used

Field Type	Operator	Process Columns
String	eq – equals eqic – equals ignoring case contains – contains startswith – starts with	host, sddUser, sasworkPath, sasworkSizeUnits, sasutilPath, sasutilSizeUnits, tags, managedServer
Integer, Double	eq – equals gt – greater than ge – greather than or equal to lt – less than le – less than or equal to	processId, elapsedTime, cpuTime, cpuPercent, memSize, memPercent, sasworkSize, sasutilSize
Date	eq – equals gt – greater than lt – less than	started

            - The value of the attribute will be used to perform the comparison.
        - <ActionBracket> - defines how the sub-element results are combined. It can contain either <ActionBrackets> or <ActionConditions>. If "not" is the logic condition for the <ActionBracket> then it will only contain one sub-element (either an <ActionBracket> or <ActionCondition>. If the logic condition is "and" or "or", then there can be multiple sub-elements.
            - @condition – the logic condition used to combine the sub-elements.
    - <Exclude> - a set of <ActionCondition> and <ActionBraket> elements that are evaluated using an "OR" operator to determine if the process meets the criteria for exclusion.

- <ActionConsequences> - the list of consequences that should be applied if the process meets the conditions.
    - <ActionConsequence> - a single consequence.
        - @type – the type of action that should be performed. See the section below on "Defining Consequences" to determine what the valid values are.

Defining Consequences
These are defined in the JC_HOME/conf/sddActions/sddConsequenceDefs.xml file. The users should not edit this file. However, the @type is the name that would be used in the sddActions.xml file.

Considerations / Assumptions:

-

Test Automation:

- JUnit tests for testing consequence completeness and the comparison logic. Also the building up of actions via the "bracket" concept.

Known Limitations:

- None