Tools, Debugging and Troubleshooting

Fundamentals of IdentityIQ Implementation

IdentityIQ 7.0

SailPoint

Overview

Tools, Debugging and Troubleshooting

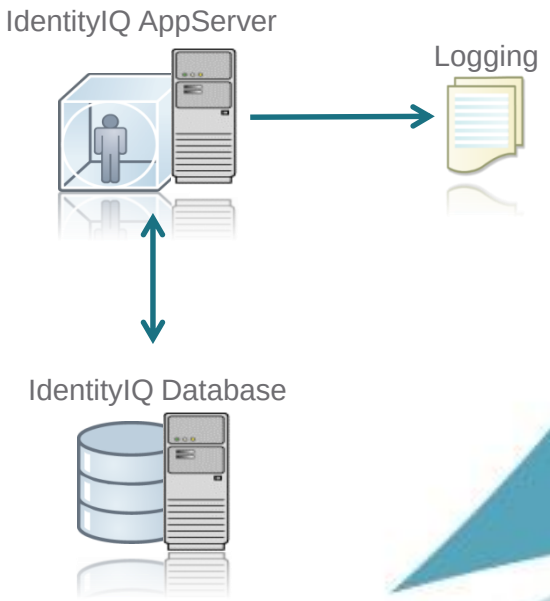
- Factors in Successful Troubleshooting & Resolution
- Tools
  - Logging, Options & Configuration
  - Console
  - Debug Page
- Best Practices for Debugging

Factors in Successful Troubleshooting

- Detail-Oriented
  - Small inconsistencies can often cause large headaches. *Infamous quote "I thought I could just ignore…"*
  - Take detailed notes, follow documentation steps carefully
- System Familiarity
  - Knowing about IdentityIQ and what is going on can make a huge difference in determining causes of issues.
  - Training and time spent with the product.
- Methodical Testing
  - Repeatable testing is the only way to guarantee success.
  - Don't change more than one variable at a time when testing
- Environmental Awareness
  - Keeping aware of the happenings on a larger scale (database, application server, JVM) will help.
  - It might not be related to IdentityIQ

Logging

- Logging is a core investigative tool.
- Logging Options
  - log4j
  - Standard Out (App Server location)
  - Email redirection
  - Audit configuration
  - Syslog logging configuration

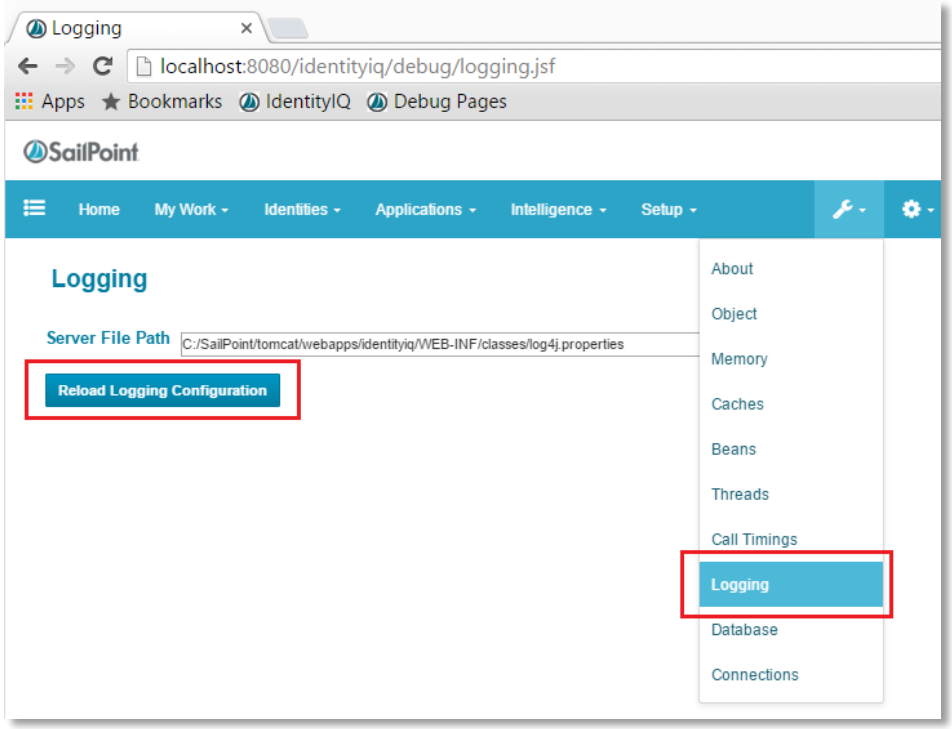


Log4J Configuration

- Log4J 101
  - Logging Levels:
    - trace
    - debug
    - info
    - warn
    - error
  - Configured in log4j.properties file
- Global Configuration
  - log4j.rootLogger=error,file (change error to other level for global log4j changes)
- Logging Configuration per Class
  - Uncomment out Class Logger names to enable.
  - Disabled:
    - #log4j.logger.sailpoint.api.Aggregator=debug
  - Enabled:
    - log4j.logger.sailpoint.api.Aggregator=debug
- Changing Logging Levels for individual classes
  - Append Logging Level to end of Class Logger
    - log4j.logger.sailpoint.Aggregator=**<logging level>**

Log4J Configuration

- **<install dir>/WEB-INF/classes/log4j.properties**
- Reload or change log file via Debug Page (preferred method)
  - Multiple log4j files for different purposes



Note: Optionally, bounce application server to reload

Log4J Example

- Inside of rule

```
log.error("This is an error message");
log.warn("This is a warn message");
log.info("This is an info message");
log.debug("This is a debug message");
log.trace("This is a trace message");
```

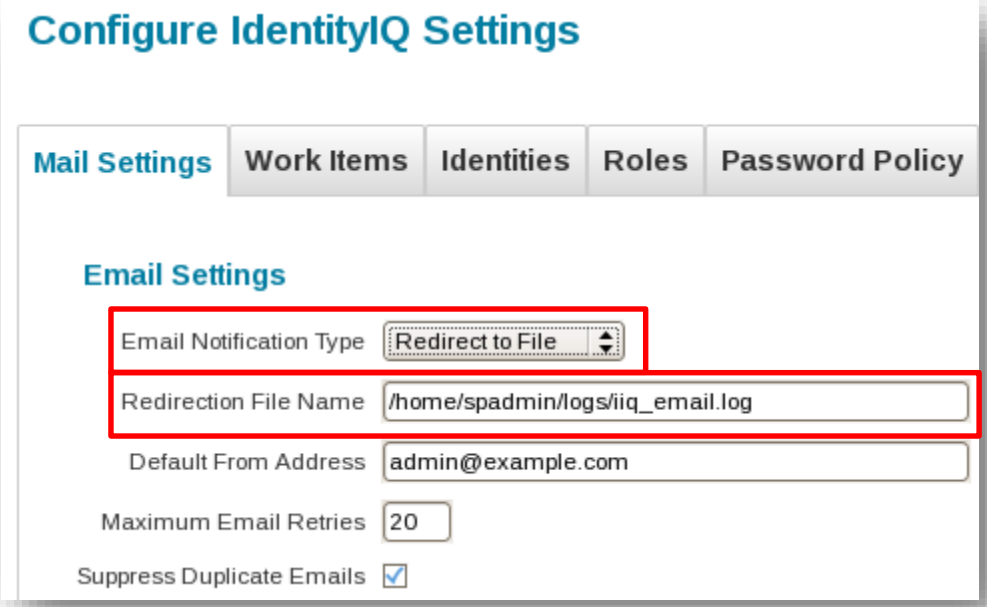
- What gets printed into log file if log level is set to "info"?

Standard Out logging

- Standard Out
  - Usage: System.out.println(“I’m logging this message.”);
- Best Practices
  - Not as useful as log4J since these messages are always printed no matter what
  - Useful for quick and dirty debugging
- Configuration
  - App server configuration determines where to send this information

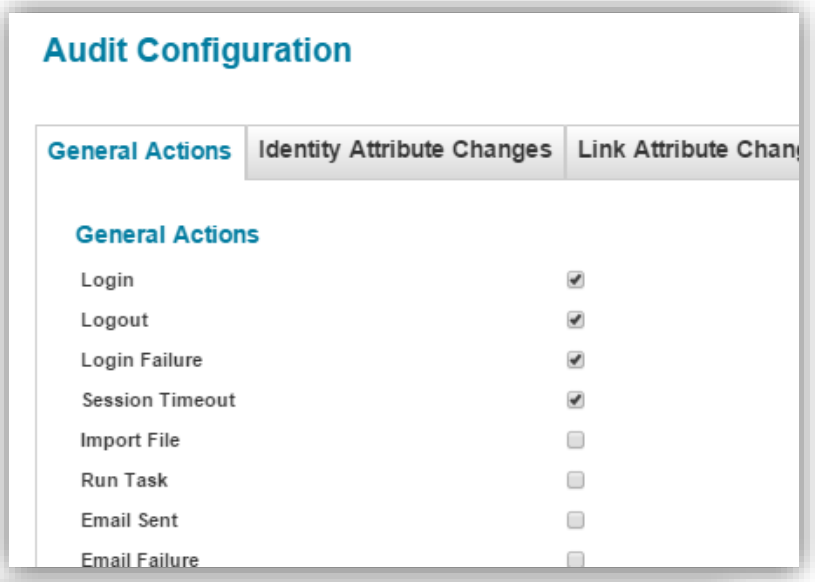
Email Logging

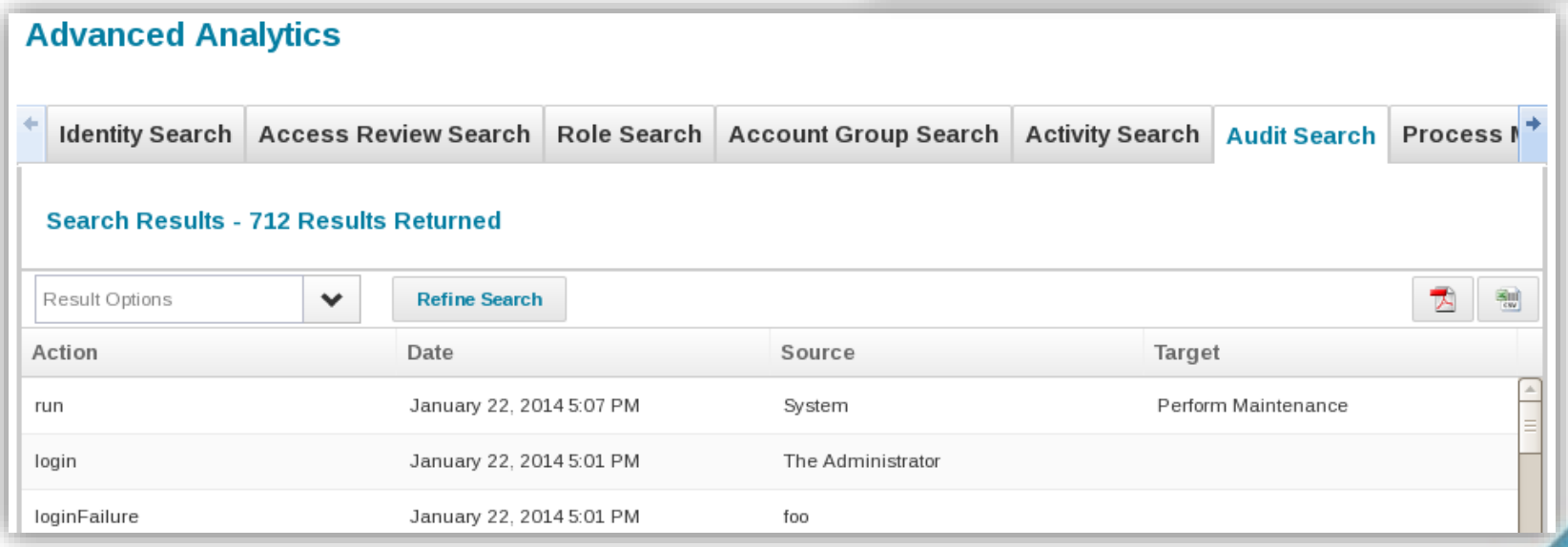
- Can redirect emails to file for testing, debugging, and troubleshooting



Auditing

- Configure
  - Gear → Global Settings → Audit Configuration
- View
  - Intelligence → Advanced Analytics → Audit Search





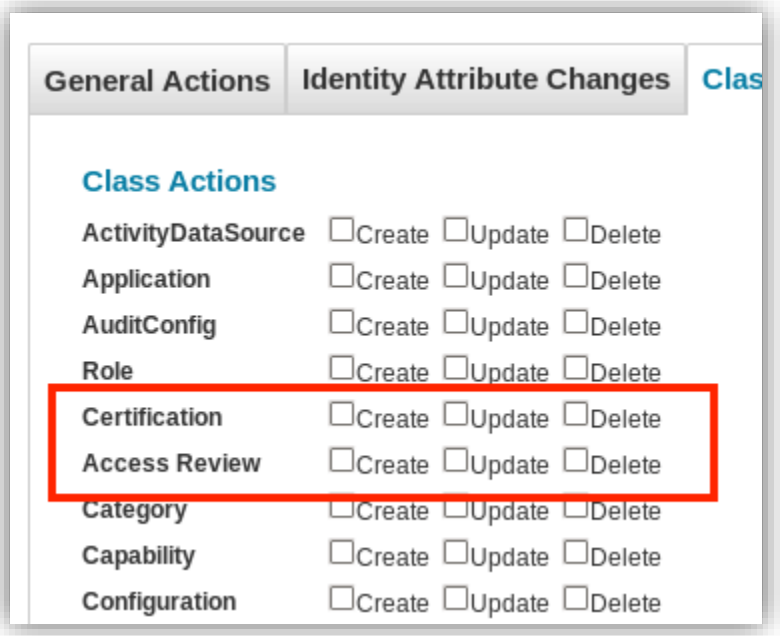
Auditing

Extending

- Add additional classes to AuditConfig

```
<AuditClass displayName="Role" name="Bundle"/>
<AuditClass displayName="Certification" name="CertificationGroup"/>
<AuditClass displayName="Access Review" name="Certification"/>
<AuditClass name="Category”/>
```

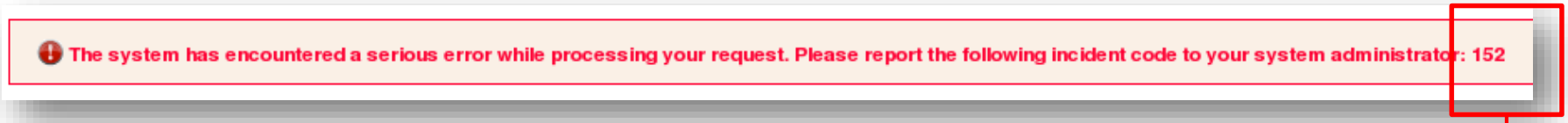
- Enable



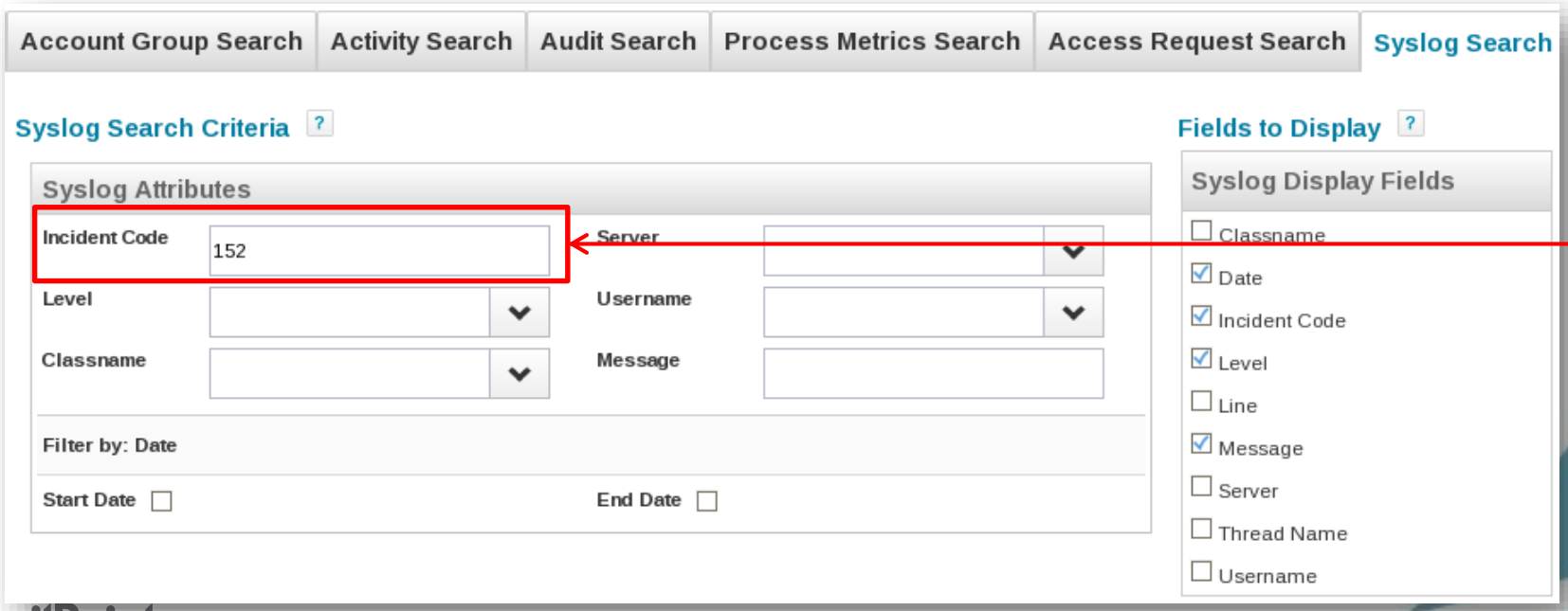
Note: Be aware of how much data you will collect

Syslog – Incident Codes

- When errors occur, an incident code may display in the UI

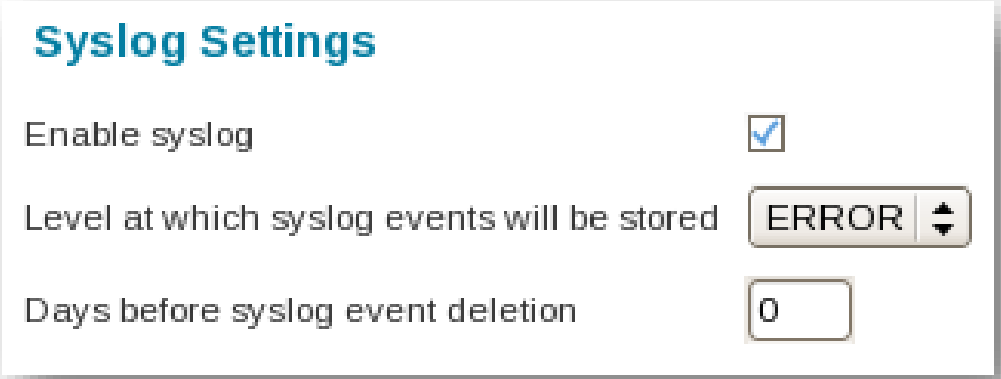


- Enter incident code to retrieve details
  - Intelligence → Advanced Analytics → Syslog Search



Syslog Log

- Redirects log4j messages to IdentityIQ database
  - One location for error messages
  - Avoid viewing host based log files
- Configuration
  - Default = enabled, with no event deletion
  - Set “Days before syslog event deletion” (best practice)
    - Typically set to 30 days
  - Gear → Global Settings → IdentityIQ Configuration → Miscellaneous



IdentityIQ Console

- Command-line driven interface
- Connects directly to database
  - Can be used to troubleshoot connectivity problems
- “Quick Glance” view of what is happening
- Some commands are only available via console
  - SQL query interface
  - Test interface
- Authentication required to access console
  - Exception is spadmin with the admin password

Console – Connector Debug

- Available via the IdentityIQ Console only
- Iteration Features
  - Displays Application Link (Accounts) in XML
  - Accounts: connectorDebug <Application> iterate account
  - Groups: connectorDebug <Application> iterate group
- Connection Test Feature
  - connectorDebug <Application> test
- Also displays associated Rules
  - Build Map Rule
  - Merge Maps Rule
  - Map to Resource Object Rule
  - Customization Rule
- Output shows ResourceObjects just prior to Correlation and Creation

Connector Debug Output

- Output shows final Resource Objects

IIQ CONSOLE

```
    </Map>
  </Attributes>
</ResourceObject>

<ResourceObject displayName="John Conner" identity="T1T2T3" objectType="account"
>
  <Attributes>
    <Map>
      <entry key="Database Name" value="TEST01"/>
      <entry key="Last Login Date" value="07/1/2011"/>
      <entry key="Permission Group">
        <value>
          <List>
            <String>IT</String>
            <String>ADMINISTRATORS</String>
          </List>
        </value>
      </entry>
      <entry key="User ID" value="T1T2T3"/>
      <entry key="User Name" value="John Conner"/>
    </Map>
  </Attributes>
</ResourceObject>

Iterated [7] objects in [94 ms]
>
```

Console for Manipulating Objects

- list – display all object types
- list <object> - display all objects of specific type
  - list rule
  - list workflowcase
  - list workitem
  - Note: Supports wildcarding
- count <object> - display count of type of object
  - count identity
- get <object class> <name>
  - get rule “Test Rule”
- delete <object class> <name or id>
  - delete identity a*
  - delete certificationgroup *

Console for Import/Export

- export [-clean[=id,created…]] <file> [class]

  export -clean application.xml application

- checkout <class> <name> <file> [-clean[=id,created…]]

  checkout application “PAM” pam.xml –clean

- import <filename>

  import pam.xml

Other Useful Console Commands

Category	Command	Details
Rules	**list rule**	Lists all rules
	rule <rulename>	Runs a rule
Tasks	**list taskdefinition**	Lists all tasks (and reports) in the system
	tasks	Lists all scheduled tasks
	run <taskname>	Runs a task
General	**about**	Lists info about system
	source	Load and execute command file into console

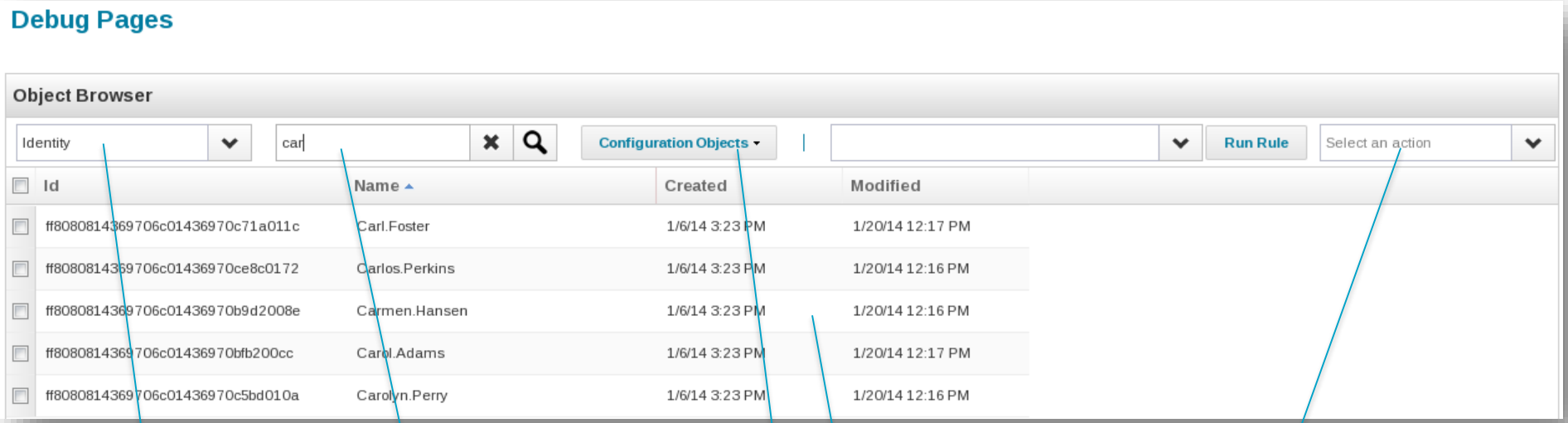
Advanced Configuration and Debugging

IdentityIQ Debug Page

- Only available to users with **System Administrator** capability
- Hidden context root for debugging options.
  - <IdentityIQ URL>/debug/
  - For Example, http://localhost:8080/identityiq/debug/
- Provides Many Features
  - Viewing of all XML Objects
  - Editing of Raw XML Objects
  - Creating and Deleting of Objects
  - Access to Configuration
    - System Configuration
    - UI Configuration
  - Memory Usage
  - Garbage Collection Methods

Debug Page – Default View

Object Viewer



Object to Search for

Creation and Modification Dates

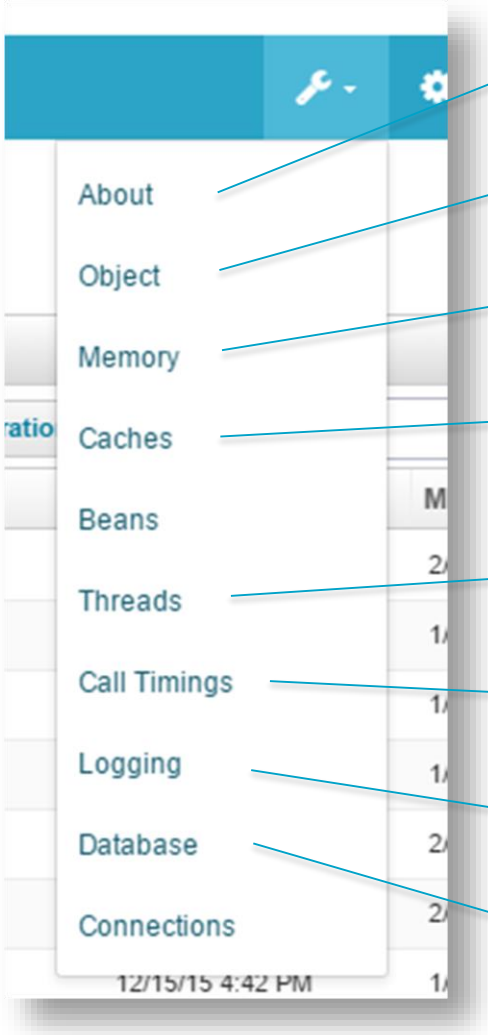
Object Type to Search for

Manipulate Configuration Objects
UIConfig
System Config
Identity Mappings

Create and Delete Objects

Debug Page

System Information



- About IIQ Environment
- Object Viewer (Default Page)
- Display Current Memory Usage
- Reset Caches
- Troubleshoot Threads
- Observe Call Timings
- Reload Log4J configuration
- DB Settings/# of Connections Used

Hints, Tips and Tricks

- Troubleshooting database issues using p6spy
  - Supports logging of all SQL queries going to the Database
  - Wraps the JDBC driver you are using with the p6spy driver
  - All queries are then logged to a specified log file

- Setup
  - Edit spy.properties
    - module.log=com.p6spy.engine.logging.P6LogFactory
    - realdriver=com.mysql.jdbc.Driver
    - logfile=/home/spadmin/logs/spy.log
    - deregisterdrivers=true

  - Edit iiq.properties
    - #dataSource.driverClassName=com.mysql.jdbc.Driver
    - dataSource.driverClassName=com.p6spy.engine.spy.P6SpyDriver

  - Stop and restart console and app server

Hints, Tips and Tricks

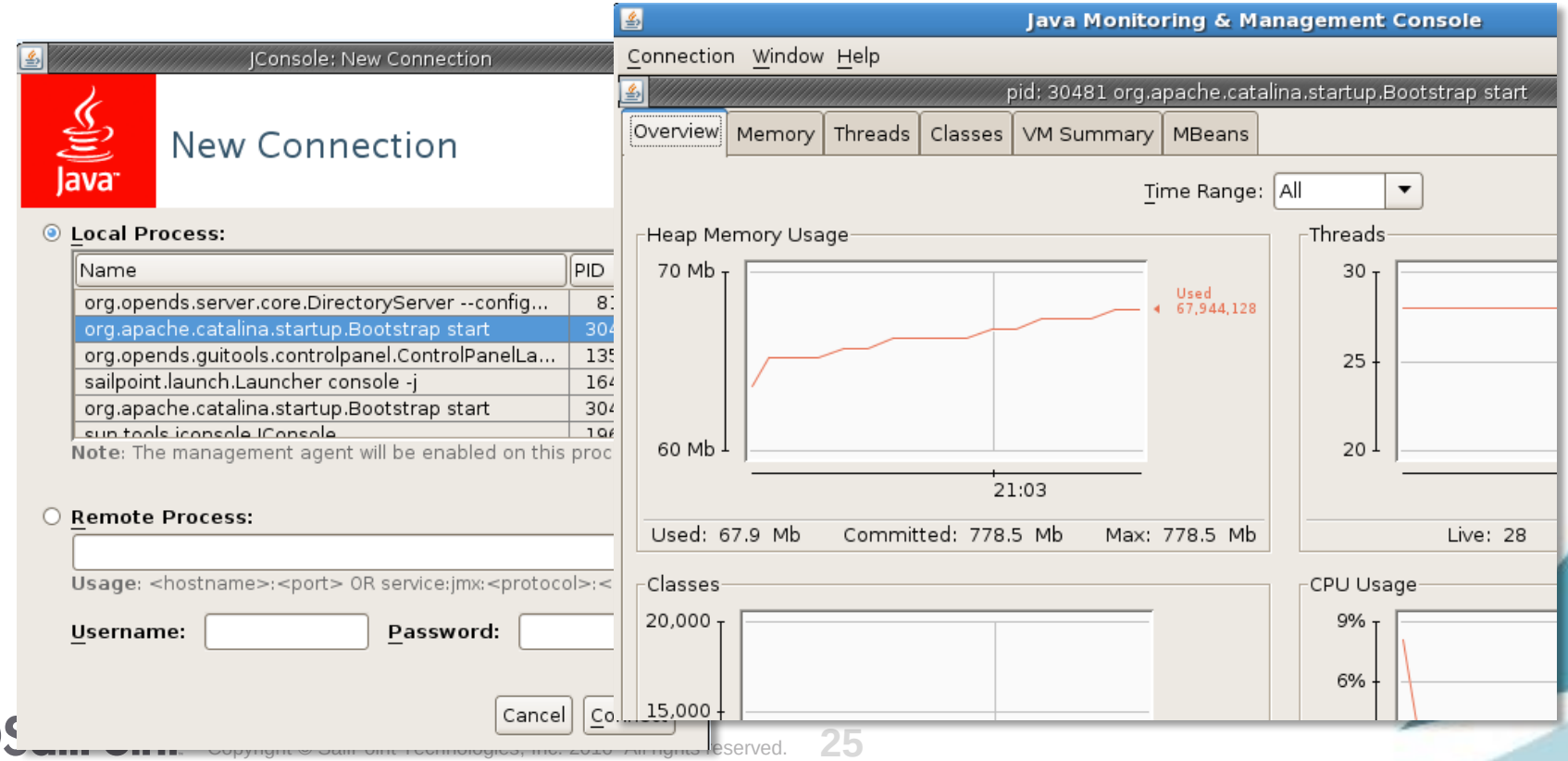
- Troubleshoot SailPoint objects using the toXml() method
  - All objects in the sailpoint.object package support the toXml() method
  - Use the following in your rule, or workflow code to output SailPoint objects and use for troubleshooting

```
Identity foo =
context.getObjectByName(Identity.class,"spadmin");
System.out.println("Identity XML = " + foo.toXml());
```

  - Useful for:
    - Determining what's available when writing rules
    - Showing the makeup of an Identity
    - Showing contents of Provisioning Plans/Projects

Hints, Tips and Tricks

- Troubleshooting Application Heap/Memory Usage
- Run jconsole (or another tool that can use JMX to monitor) to monitor resource consumption and performance of applications running on Java



Questions?

SailPoint