Reporting

Fundamentals of IdentityIQ Implementation

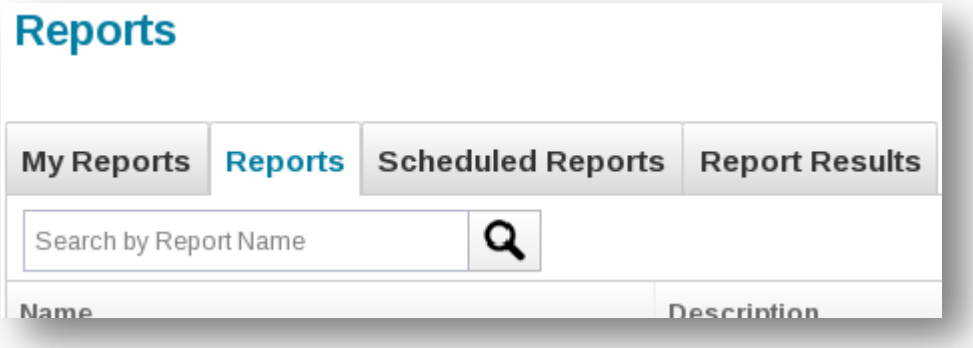
IdentityIQ

Overview

Reporting

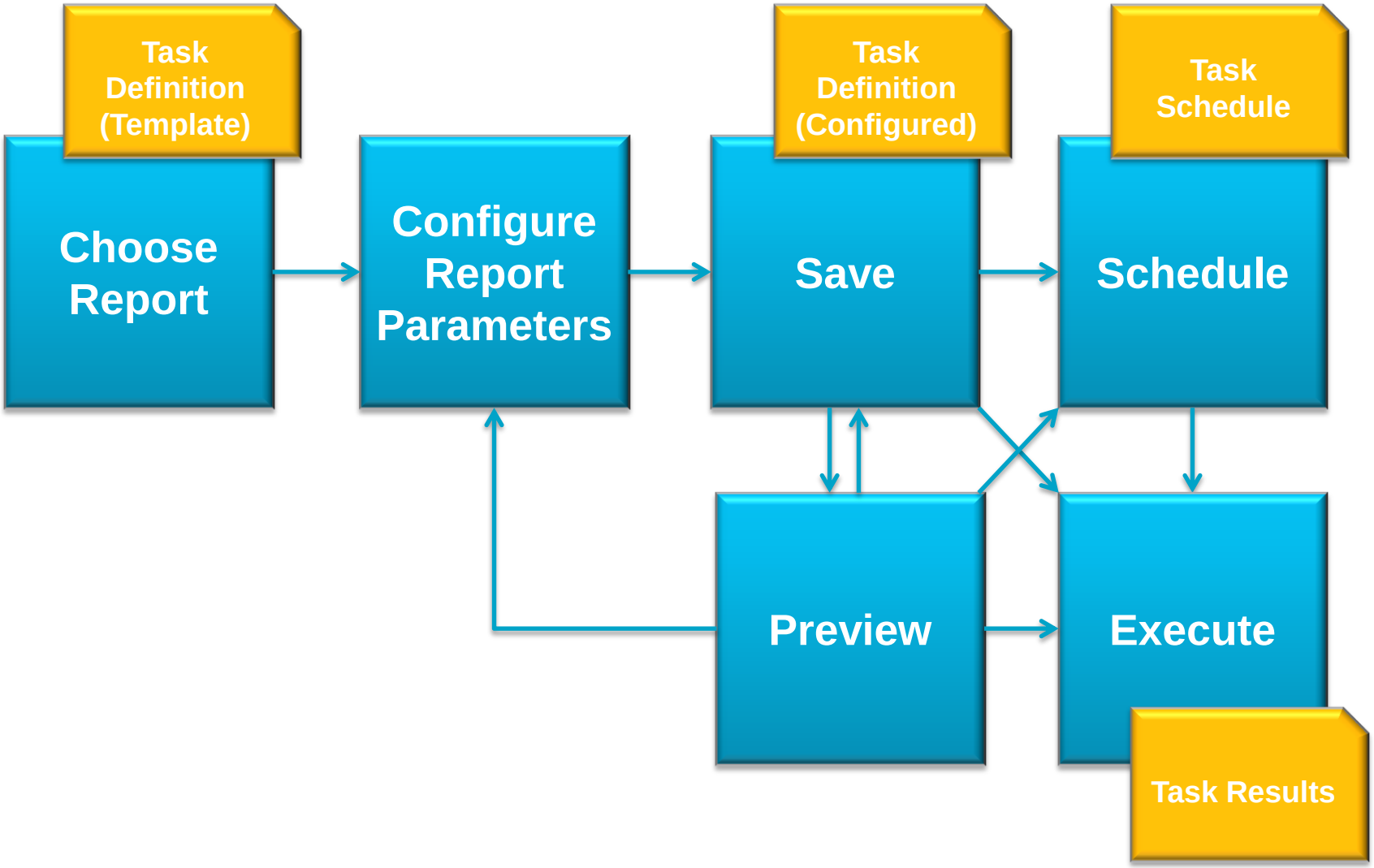
- Understand the reporting architecture of SailPoint IdentityIQ
- Learn about creating custom reports including
  - Defining Data Sources
  - Report Column Configuration
  - Query Configuration

Reporting Overview



- Reports Tab
  - Report Templates
    - Used for creation of reports
- My Reports Tab
  - Saved Reports configured from Report Templates
    - Represent saved configurations of report templates
- Scheduled Reports Tab
  - Details information about report schedules
- Report Results Tab
  - Results of ad-hoc or scheduled reports (Results and links to PDF/CSV)

General Reporting Flow



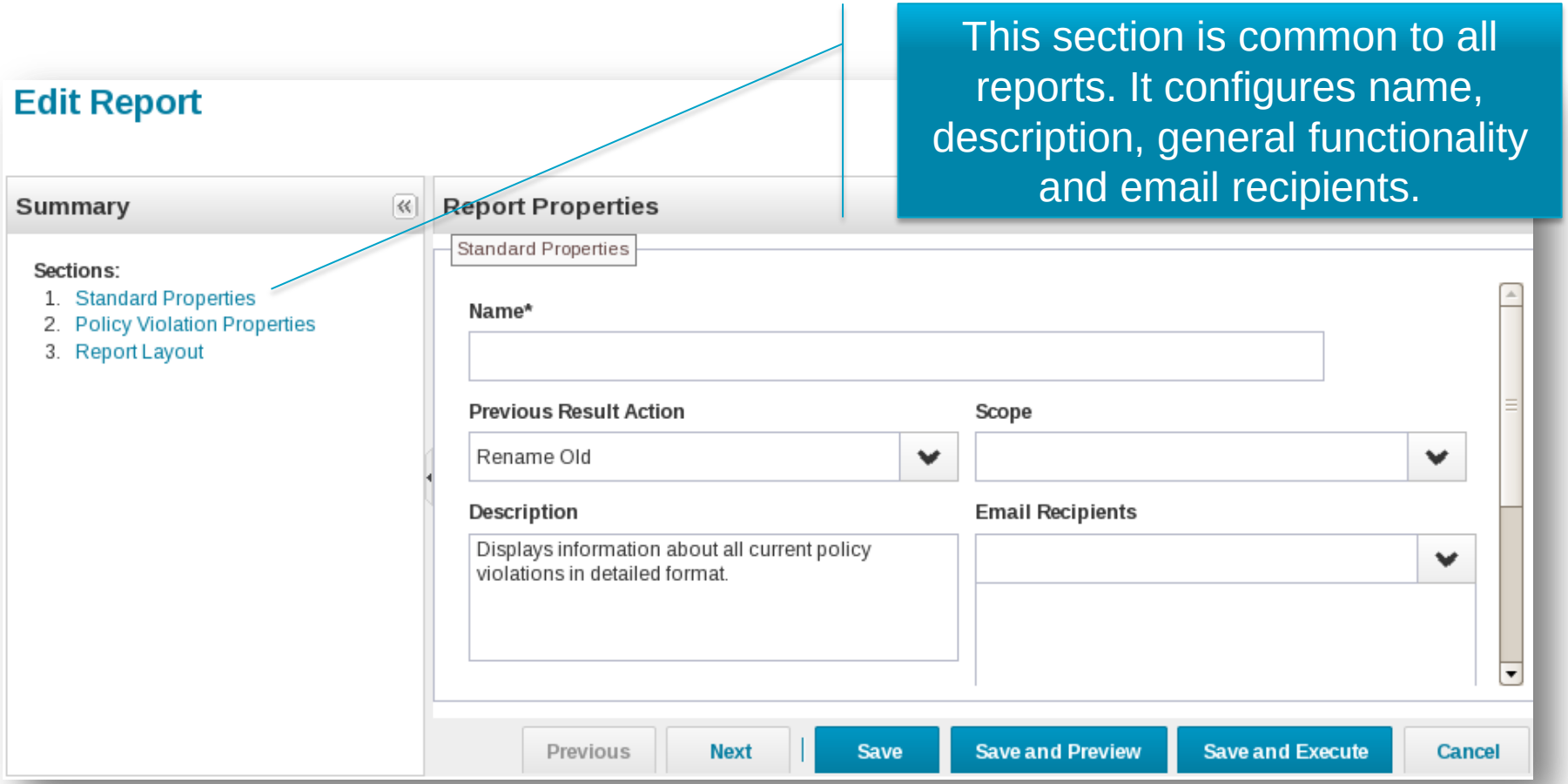
Configuration of Report Parameters

- Configuration specifies
  - Standard properties
    - Name
    - Description
    - Email recipients
  - Data filtering
    - By date range
    - By application
    - By manager
    - etc…
  - Column layout (adjustable in preview as well)
    - Which columns are visible
    - Sort by
    - Group by

Reporting Configuration

Sections

- All reports have three main sections

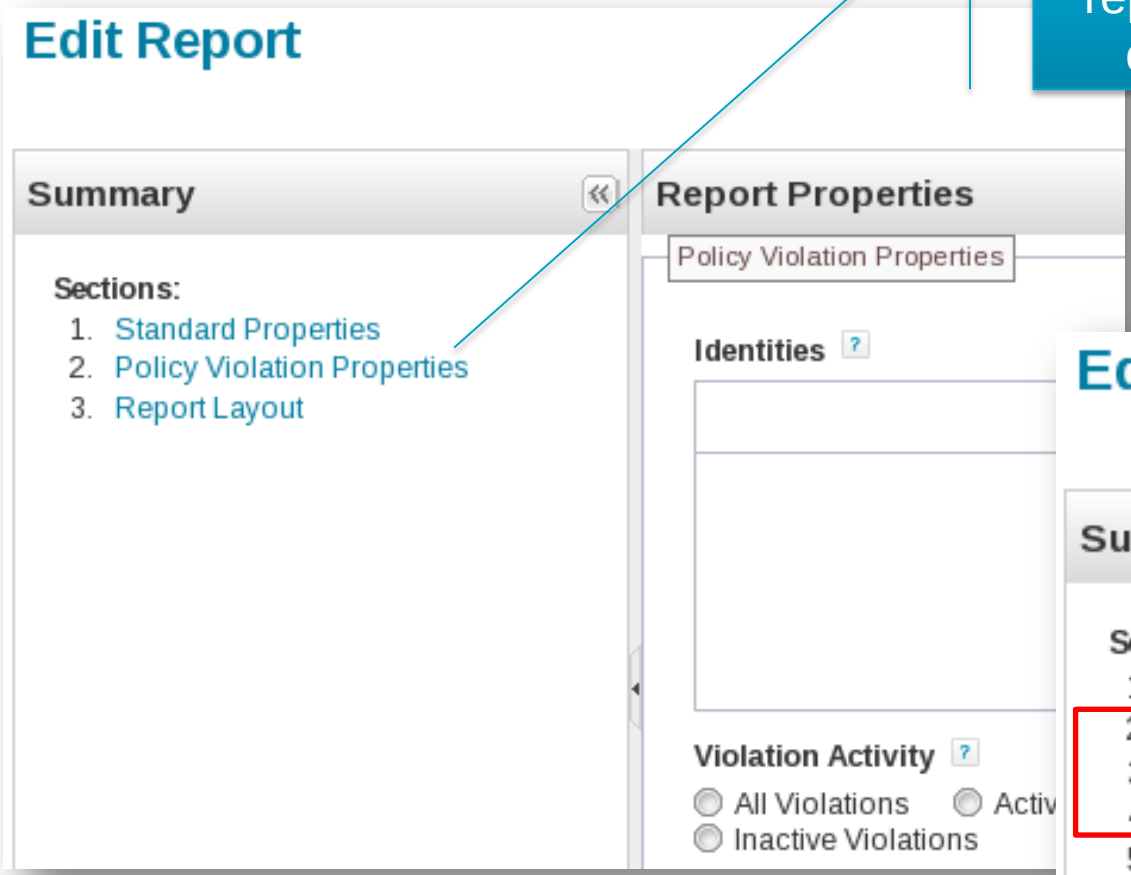


Reporting Configuration

Report Specific

- Middle section is report specific

This middle section varies for each type of report. For custom report templates, this section is created using Form objects

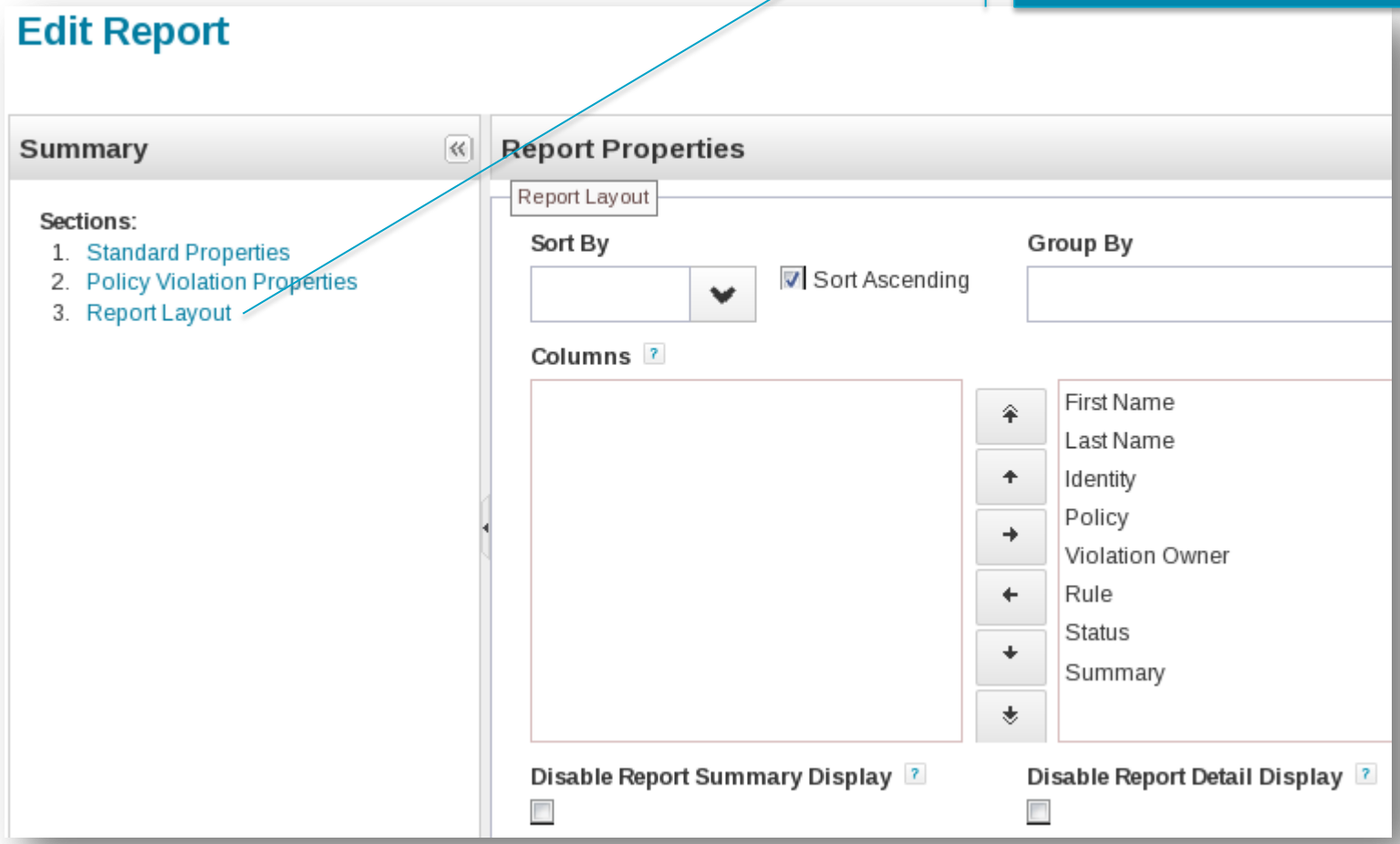


Reporting Configuration

Layout

- Report Layout (Grid Control)

Sort By, Group By, Columns, Summary and Detail display options



SailPoint

Developing New Custom Reports

- Reporting architecture allows for Implementers to extend IdentityIQ to create new reports
- Every Report Definition is just a specialized Task
  - Reports inherit IdentityIQ Task characteristics
    - Configuration (saving configured Task Definition objects)
    - Schedules (TaskSchedule objects)
    - Results (TaskResult objects)
  - Report definitions define
    - Summary and Charting options
    - What object type to report on (Identity, Policy Violation, etc.)
    - Report Layout (Columns and ordering to display)
    - Query to retrieve supporting data
      - SailPoint Filter, Java Datasource, HQL

New Reporting – Implementation Details

- New Report Executor
  - Java class:
    - sailpoint.reporting.LiveReportExecutor
  - LiveReportExecutor uses the XML report definition encapsulated within the TaskDefinition to generate the report
- Creating New Custom Reports
  - Existing Live Reports can be used as examples for custom reports
  - Generate new Task Definition including
    - Configuration Options
    - Query to execute
    - Columns to retrieve

Report XML Overview

Policy Violation Report

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE TaskDefinition PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<TaskDefinition created="1346692200540" executor="sailpoint.reporting.LiveReportExecu
  <Attributes>
    <Map>
      <entry key="report">
        <value>
          <LiveReport title="Policy Violation Report">
            <Chart category="status" groupBy="status,policyName" series="policyName"
            <DataSource objectType="PolicyViolation" type="Filter">
            <ReportForm> ... </ReportForm>
            <Columns> ... </Columns>
            <ReportSummary title="rept_cert_summary_title"> ... </ReportSummary>
          </LiveReport>
        </value>
      </entry>
    </Map>
  </Attributes>
  <Description> ... </Description>
  <RequiredRights> ... </RequiredRights>
  <Signature> ... </Signature>
</TaskDefinition>
```

- LiveReport
- Chart Definition
- DataSource Definition
- Form Reference
- Column Configuration
- Report Summary

Data Query

- Start by thinking of the construct of a basic SQL statement

  select <columns> from <table> where <criteria>;

  Example:

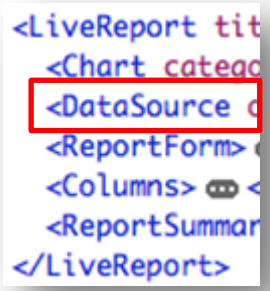
  select id, name from spt_identity where name = 'Adam.Kennedy';

  columns *table* *criteria*

- In our reporting engine, the mappings for each item are:

<columns>	Column Configuration
<table>	SailPoint Object
<criteria>	Query (Defined as Filter, HQL or Java DataSource)

Example – Policy Violation Report



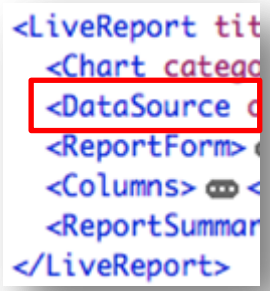
- What to search for?
  - Defined as objectType and type

    <DataSource objectType="PolicyViolation" type="Filter”>

  - We will search for Policy Violation objects using a Filter

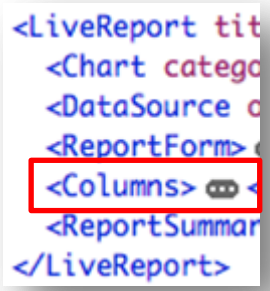
Example – Policy Violation Report

- What Query to Run?
  - Default behavior is to bring back all objects of the type specified.
  - Query is configurable
    - Data Source
    - Query Parameters
    - Custom Forms




```
<DataSource objectType="PolicyViolation" type="Filter">
    <QueryParameters>
        <Parameter argument="identities" property="identity.id"/>
        <Parameter argument="status" property="active">
            <ValueScript>
                …
            </ValueScript>
        </Parameter>
        <Parameter argument="policies" property="policyId"/>
        <Parameter argument="violationDate" operation="LT" property="created"/>
        <Parameter argument="actualStatus" property="status"/>
    </QueryParameters>
</DataSource>
```

Example – Policy Violation Report



- What columns to retrieve?
- Defined as Columns and ReportColumnConfig Items in XML
- Lists the items that we need to display defined by properties

```
<Columns>
        <ReportColumnConfig field="firstname“ header=”FirstName"
property="identity.firstname" sortable="true" width="110"/>
        <ReportColumnConfig field="lastname" header=”LastName"
property="identity.lastname" sortable="true" width="110"/>
        <ReportColumnConfig field="name" header=”Name"
property="identity.name" sortable="true" width="110"/>
    …
```

Example – Policy Violation Report

Signature and ReportForm

- Signature defines filters which can be specified by report user

```
<Signature>
   <Inputs>
      <Argument multi="true" name="identities" type="Identity">
            <Description>rept_input_violation_report_identities</Description>
     </Argument> …
   </Inputs>
 </Signature>
```

- Optional ReportForm can be specified to manage UI presentation of filters

```
<ReportForm>
     <Reference class="sailpoint.object.Form" id="ff8080814528eeb90854981201b4"
      name="Policy Violation Report  Form"/>
</ReportForm>
```


SailPoint

Direct Connect – Data Integration

- In 6.0, we flattened a good bit of the data model to makes it easier to get at the data
  - Recommended approach is to replicate the database (or a portion of the database)
  - Use off-the-shelf Business Information (BI) tools to query the data
- IIQ Console command
  - *iiq exportviews*
  - Creates views for exported data (supported mappings for extended attributes)
    - Example: “extended1” becomes “division” in view

Note: Data Export task will remain for backwards compatibility.

SailPoint

Questions?

Exercise Preview

Section 3, Exercises 6

- Exercise 6: Creating and Extending a Custom Report
  - Load custom report from XML file
  - Investigate the report and modify its layout
  - Extend the report to display additional data
- Optional Extension Exercises
  - Extend the report using signature and forms
  - Extend the report to limit returned data

Report Exercise Preview

- /home/spadmin/ImplementerTraining/config
  - Report-CustomCapabilities.xml
    - **Initial custom report – no parameters**
    - **Provides all identities and their capabilities**
  - Report-CustomCapabilities2.xml
    - Extended report
    - Added “Region” and “Location”
  - Report-CustomCapabilities3.xml
    - Signature example
    - Filter by “Identity “and “Manager”
  - Report-CustomCapabilities4.xml
    - Form example
    - Format the data input section: title and format
    - Uses Report-CapabilitiesForm.xml
  - Report-CustomCapabilities5.xml
    - Added query option to show only capabilities with values