What is an IntegrationConfig?

IntegrationConfig objects define the interaction between IIQ and external applications for provisioning requests and role synchronization.   Each IntegrationConfig defines the java class that will execute each request, the applications managed and the settings that define the behavior of the integration.

## **Integration Config vs Before Provisining Rule**

The best way to describe a integration config is the configuration of an integration module. In turn the integration module is a write-only connector, that can be used to redirect provisioning to another system: a ticket system, a third party provisioning engine, an intermediate table used for inter-system communication, debug modules for testing provisioning, etc. The integration module is an older mechanism than most of the current direct connectors, which can use the same channel to read and write information. The integration rule is often used to process the provisioning plan before it is handed to the integration module. The before provisioning rule can also do that, but is not executed when an integration module is used for provisioning.

In some cases, you may want to use a before provisioning rule to call an integration module. As an example, a customer wanted to handle account creation and deletion manually in the first phase, but let IdentityIQ only update attributes and entitlements of existing accounts. We called the ticketing integration from the before provisioning rule and stripped the plan (emptied it) in case of creation or deletion. This would cause both the before provisioning and integration rule to be executed.

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.util.List;

**import** sailpoint.api.SailPointContext;

**import** sailpoint.connector.Connector;

**import** sailpoint.integration.AbstractIntegrationExecutor;

**import** sailpoint.object.Application;

**import** sailpoint.object.IntegrationConfig;

**import** sailpoint.object.ProvisioningPlan;

**import** sailpoint.object.ProvisioningPlan.AccountRequest;

**import** sailpoint.object.ProvisioningResult;

**import** sailpoint.object.Rule;

**import** sailpoint.object.Schema;

**public** **class** ProvisionIntegrationModule **extends** AbstractIntegrationExecutor{

**private** SailPointContext context;

@Override

**public** **void** configure(SailPointContext context, IntegrationConfig config)

**throws** Exception {

// **TODO** Auto-generated method stub

**this**.context = context;

**super**.configure(context, config);

}

@Override

**public** ProvisioningResult provision(ProvisioningPlan plan) **throws** Exception

{

Application object = context.getObject(Application.**class**, "ACME Fin");

System.***out***.println("acme fin applicaton--"+object.toXml());

Rule rule = **this**.context.getObject(Rule.**class**,"test AfterProvisioning");

context.runRule(rule, **null**);

// **TODO** Auto-generated method stub

/\* System.out.println("Plan in integration config:"+plan.toXml());

List<AccountRequest> accReqs = plan.getAccountRequests();

for(int i=0;i<accReqs.size();i++)

{

AccountRequest accReq = accReqs.get(i);

String applicationName = accReq.getApplicationName();

ProvisioningPlan.AccountRequest.Operation operation = accReq.getOperation();

System.out.println("App Name:"+applicationName);

Application app = this.context.getObjectByName(Application.class,applicationName);

System.out.println("1---"+operation);

String filePath = (String)app.getAttributeValue("file");

System.out.println("2");

if(operation != null && operation.equals(ProvisioningPlan.AccountRequest.Operation.Create))

{

Schema sch = app.getSchema(Connector.TYPE\_ACCOUNT);

List<String> appAttrs = sch.getAttributeNames();

StringBuffer lineItem = new StringBuffer();

for(int j=0;j<appAttrs.size();j++)

{

lineItem.append((String)(accReq.getAttributeRequest(appAttrs.get(j)).getValue()));

lineItem.append(",");

}

String line = lineItem.substring(0, lineItem.length()-1);

System.out.println("Line Item:"+line);

writeLine(filePath,line);

}

}\*/

ProvisioningResult pr = **new** ProvisioningResult();

pr.setStatus(ProvisioningResult.***STATUS\_COMMITTED***);

**return** pr;

//return super.provision(plan);

}

**private** **void** writeLine(String filepath,String line) **throws** Exception

{

File file = **new** File(filepath);

FileWriter fw = **new** FileWriter(file,**true**);

BufferedWriter bw = **new** BufferedWriter(fw);

bw.newLine();

bw.write(line);

bw.close();

fw.close();

System.***out***.println("Line has been written successfully!!");

}

}

**IntegrationConfig object**

<IntegrationConfig executor="com.example.integration.ProvisionIntegrationModule" name="ProvisioningIntegrationModule" roleSyncStyle="none">

<Attributes>

<Map>

<entry key="operations" value="Create,Modify"/>

</Map>

</Attributes>

<ManagedResources>

<ManagedResource>

<ApplicationRef>

<Reference class="sailpoint.object.Application" name="TARGET"/>

</ApplicationRef>

</ManagedResource>

</ManagedResources>

</IntegrationConfig>

**Integration rule:**

**<?xml version='1.0' encoding='UTF-8'?>**

**<!DOCTYPE Rule PUBLIC "sailpoint.dtd" "sailpoint.dtd">**

**<Rule created="1589539658272" id="4028698172173fae017217f25a1f0037" language="beanshell" modified="1589539915160" name="integration rule" type="Integration">**

**<Signature>**

**<Inputs>**

**<Argument name="log"/>**

**<Argument name="context"/>**

**<Argument name="identity"/>**

**<Argument name="integration"/>**

**<Argument name="plan"/>**

**</Inputs>**

**</Signature>**

**<Source>**

**System.out.println("in integration rule plan"+plan.toXml());**

**System.out.println("in integration rule identity"+identity.toXml());**

**System.out.println("in integration rule integration"+integration.toXml());**

**</Source>**

**</Rule>**