

Population Automation

A Successful BSM Depends on an Accurate CMDB

By Bill Tobey

If you're implementing [Business Service Management \(BSM\)](#) for any of the usual reasons—to increase the reliability and availability of your critical business services, to improve your service delivery performance, or to reduce the costs of service delivery—be aware that your success in achieving any of these objectives will be highly dependent on the quality and timeliness of the data in your Configuration Management Database (CMDB).

Today's CMDB is operational by definition. When an outage occurs, you can't do fast root-cause and impact analysis unless your CMDB can access and supply near real-time information about all services and configuration items (CIs) under management, the relationships between them, and their current service states. The CMDB is a special-case database that must provide four critical functions creating the Configuration Management System (CMS):

- Federation – Direct linkage to multiple data sources
- Reconciliation – Automated data coalescence and matching across sources to eliminate duplication
- Visualization – Automatic detection and graphic representation of the hierarchical relationships between physical and logical CIs
- Synchronization – The ability to immediately capture changes in the physical and logical infrastructure, preserving and providing a single version of the truth across all integrated systems

One of the key challenges in creating and maintaining a CMDB is the problem of populating it from the various system, asset and network management solutions where information about the IT environment originates. Manual processes may offer a high level of initial accuracy, but in large environments, their labor intensity virtually guarantees decreasing accuracy over time.

> **SCM: Population Propagation, the Automated Way**

But if you're implementing a Novell CMDB, you've got an integrated population automation solution sitting right under your right mouse button, ready to launch. The Service Configuration Manager functions as an intermediary between the CMDB and the adapters that integrate it with management applications and configuration information repositories, providing the Configuration Management System (CMS) automation. The Service Configuration Manager feature dynamically generates new element hierarchies from multiple sources. It automates CMDB synchronization to optimize data quality and reliably capture change in the IT infrastructure in near real time.

The Configuration Management System provides:

- Integration into tools that detects dependencies, characterizes normal baselines, and identifies both scheduled and unscheduled changes
- Integrated mapping of element relationships across an enterprise IT environment
- Integration of asset, configuration and change data from multiple sources into a global CMDB
- Automatic creation and dynamic maintenance of Business Service Views that eliminate manual modeling, mapping and maintenance.

CMS not only automates many of the most time- and labor-intensive aspects of configuration data management, it lets you create and store customized business rules, in the form of JavaScripts, to govern key processes. Let's look at an example.

> **From Many Sources, One CMDB**

The fundamental use case in CMDB implementation is populating the configuration data store from a large number of diverse management systems and data repositories across the environment, and potentially from other siloed CMDBs. Let's assume that you're starting such a project using the Novell Configuration Management System. You have the Novell engine up and running on a server, along with adapters for each of the management systems you'll need to integrate with. You also have the Service Configuration Manager administrative client running on a desktop or notebook. You want to begin federating basic information about configuration items—services and all the infrastructure elements that participate in their delivery—in the CMDB.

- Begin in the Java console by drilling down into the CMS service model. Select the CI container inside of a community—servers for instance—and right click. From the list of options that is presented, choose Service Configuration Manager (SCM). Then choose to create an SCM job.
- Service Configuration Manager will now display a list of the available source system adapters. Since we're seeking information on servers, select all the adapters for target systems that are likely to contain identity, role, configuration or service state information on server systems.
- Now it's time to decide what information we'll select from each of the available sources, and how we'll load that information into the CMDB. Service Configuration Manager provides a checkbox to populate the CI Attributes from the data sources, but in most cases, unless you have an exact naming match between the source attribute names and the destination CI attribute names, you will need to set up a script to do the actual processing. Regardless, the java script approach provides additional control over how CI Attributes are populated such as pre-processing to do validity checks.

To manage the selection, linking and reconciliation of CI attributes, SCM gives us the option of using a custom JavaScript, like the following:

```
// This script is called for different types of matches. It checks to see what elements exist
// in our source systems, and that each element has been accurately created in the CMDB.
// It begins with three conditions for execution.
if( this.source && element )
{
  // Define attributes (properties) that are on the source system
  if( !state.sourceAttrs )
    state.sourceAttrs = [ 'filename', 'name', 'ip_address' ]

  // Define the attributes (in matching order) that you want the source to populate
  if( ! state.destinationAttrs )
    state.destinationAttrs = [ 'Filename', 'DB Name', 'TCP/IP Address' ]

  //Using state variables above reduces memory utilization on the server.
  //Since this script is run over and over again for each matched element,
  //it reduces run time by not redefining these variables every time.
  //Use a for loop to traverse through the identified source attributes
  for( var i = 0; i < state.sourceAttrs.length; ++i )
  {
    // attempt to pull source attribute
    var value = source[ state.sourceAttrs[i] ]

    // if it exists
    if( value )
      element[ state.destinationAttrs[i] ] = value
    // then on the line above, we then set the element
    // attribute to the value of the source attribute
    // in turn, this is then persisted to the configStore (database)
  }
}
```

Additional information on scripting for SCM control can be found in the FormulaScript Guide that is included in the CMDB installation documentation.

> A Script for Accurate, Efficient, Cost-Effective Service Management

The built-in automation capabilities of the Novell Configuration Management System combined with Service Configuration Manager's support for customized JavaScript control take many of the complexity, labor intensity and data quality challenges out of creating and maintaining a configuration management database. They're just one instance of the extensive automation capabilities that help Novell BSM solutions deliver faster time-to-value, lower implementation and maintenance costs, greater accuracy, improved business value and operational performance from all services under management. To learn more visit www.novell.com/bsm.