# **Configuration Management Data Base – CMDB**

**Description**

The Configuration Management process is the method for identifying, maintaining, controlling, and verifying IT assets and managing how they connect hierarchically to form business systems. The main goal of Configuration Management is to enable effective service delivery by having an accurate record of all Configuration Items (Cis) in the Configuration Management database (CMDB). The CMDB module on the Service Central platform is the preferred tool for IT Configuration Management in Lockheed Martin Enterprise IT (EIT)

A Configuration Item (CI) is a component, service or asset that needs to be managed to provide an IT service. By providing relationships between the CIs, IT personnel can understand and thus make informed decisions based on the service(s) they underpin. Configuration Management also helps provide quicker resolution to Incidents by providing the right Cl information to support personnel. Relationships further help support personnel quickly identify potential root causes of a service outage.

NOTE: All ITIL Users have the ability to view Configuration Item records in the CMDB. In order to modify a record in the CMDB, a user must already have the General ITSM user role (elevated SC access, backend, fulfiller, etc) and have the CMDB Librarian role assigned to them.

# **CMDB Implementation Best Practices and CMDB CI Class Manager**

**Step 1: Automated Hardware Discovery & Inventory Management**

* 1. **Set Up Discovery Tools for Hardware Detection**
* **Objective**: Automate hardware discovery and inventory management.
* **Actions**:
  + Implement **network discovery tools** (e.g., Nmap, Lansweeper, or custom scripts) that scan the network for new devices and identify active hardware (servers, switches, routers, workstations, etc.).
  + Integrate these tools with the **Central Service** and **ITSM** systems to ensure all new hardware is registered automatically as it is discovered.

**1.2 Integrate Discovery Tools with CMDB**

* **Objective**: Ensure that the discovered hardware is added to the CMDB.
* **Actions**:
  + Develop a script or automation workflow that uses the **API** of the discovery tool to push hardware data into the CMDB.
  + Ensure data fields in the CMDB are populated with the necessary details like **hardware type, manufacturer, model, serial number, IP address, location, and owner**.
  + If an asset is already in the CMDB, check for discrepancies or updates and make adjustments as needed.

**1.3 Automated Asset Registration Process**

* **Objective**: Automate asset registration and updates.
* **Actions**:
  + When new hardware is discovered, create an automation rule that adds it to the **ITSM system** and triggers an asset creation in the CMDB.
  + Use **event-driven automation** that triggers the creation of an asset record when a new device joins the network.
  + Implement custom templates for asset creation to standardize the data entry in both the **Central Service** and **ITSM**.

**Step 2: Data Validation and Standardization**

**2.1 Data Validation Automation**

* **Objective**: Ensure the data in the CMDB is accurate and consistent.
* **Actions**:
  + Develop validation rules using the **API** of the CMDB to check for common data errors (e.g., missing serial numbers, duplicate entries, inconsistent naming conventions).
  + Automatically flag incomplete or erroneous data for review by the appropriate personnel.
  + Set up automated reports or dashboards that summarize any data discrepancies across hardware records for review.

**2.2 Automate Standardization**

* **Objective**: Enforce consistent naming conventions and data standards.
* **Actions**:
  + Implement a centralized configuration that dictates standardized **naming conventions**, **categories**, and **tags** for hardware assets.
  + Develop an automation script that checks asset entries and automatically applies standard naming conventions (e.g., asset type + location + serial number).

**Step 3: Real-Time Sync and Automated Updates**

**3.1 Automated Syncing with ITSM and Central Service**

* **Objective**: Ensure the CMDB remains up-to-date by syncing with ITSM and Central Service tools.
* **Actions**:
  + Integrate the CMDB with **ITSM** and **Central Service** systems using APIs.
  + Set up **real-time synchronization** that automatically updates the CMDB whenever a change is made in the **ITSM system** (e.g., when a hardware asset is decommissioned or updated in ITSM).
  + Implement event-based triggers to automatically pull in hardware updates from these tools (e.g., when an ITSM ticket is closed for a hardware asset, update the CMDB).

**3.2 Scheduled Synchronization for Continuous Updates**

* **Objective**: Regularly update the CMDB to reflect changes in hardware.
* **Actions**:
  + Set up **scheduled tasks** (e.g., daily, weekly) to pull new data from ITSM and Central Service into the CMDB.
  + Set up a **reconciliation process** that compares CMDB data against ITSM records, correcting discrepancies and ensuring alignment between both systems.

**Step 4: Change Management Integration**

**4.1 Integration with Change Management Processes**

* **Objective**: Automate hardware change processes to update the CMDB.
* **Actions**:
  + Integrate CMDB with the **Change Management** module of ITSM to automatically log any hardware changes, such as upgrades, replacements, or retirements.
  + When a change request for hardware (e.g., replacing a server or upgrading a router) is approved in ITSM, trigger an automation that updates the hardware record in the CMDB.
  + Use workflows to update asset statuses in real-time based on ITSM ticket resolution (e.g., hardware decommissioned).

**4.2 Automate Approval Workflows**

* **Objective**: Ensure hardware-related changes follow the formal process and are automatically reflected in the CMDB.
* **Actions**:
  + Implement **approval workflows** within ITSM for hardware changes (e.g., hardware upgrades or decommissioning).
  + Once approved, trigger an automation that updates the CMDB, marking the hardware as “decommissioned” or “upgraded,” and making the corresponding changes to asset attributes like **location** or **status**.

**Step 5: Audit and Reporting Automation**

**5.1 Automated Audits of Hardware Inventory**

* **Objective**: Ensure the CMDB is consistent with the actual hardware infrastructure.
* **Actions**:
  + Schedule **automated audits** that compare the CMDB records with actual hardware. For example, using **agent-based or network discovery tools** to cross-reference what’s in the CMDB with what’s currently active on the network.
  + Set up notifications or alerts when discrepancies are detected, so that manual checks can be performed.

**5.2 Automated Reporting**

* **Objective**: Provide stakeholders with regular reports and insights.
* **Actions**:
  + Develop automated reports that summarize the state of hardware assets, including hardware inventory levels, lifecycle stages, and upcoming maintenance or replacement needs.
  + Use **reporting tools** integrated with the CMDB and ITSM to generate regular, scheduled reports (e.g., weekly, monthly) on hardware status, health, and upcoming changes.

**Step 6: Continuous Improvement and Monitoring**

**6.1 Feedback Loop and Process Improvements**

* **Objective**: Continuously improve the automation process and system integrations.
* **Actions**:
  + Collect feedback from system users (e.g., asset managers, IT support teams) to identify pain points or areas for improvement.
  + Use **monitoring tools** to track the performance of the automation workflows and make adjustments based on failures or inefficiencies.

**6.2 Enhance and Scale Automation**

* **Objective**: Scale and improve automation as the infrastructure grows.
* **Actions**:
  + Regularly evaluate new **automation tools** or scripts that can enhance the process (e.g., more granular device discovery, better error detection, or improved integration with other internal systems).
  + Scale the automation process as the LMI infrastructure grows, ensuring that the CMDB and associated processes can handle increased volumes of hardware.

## **CMDB Data View**

### Change Traceable:

* Every change to CMDB data is fully traceable and logged by date and time.
* For example: Source of data such as Manual, Discovery, SCCM, any third party tool, user, etc., which includes the new and previous value.

## **Query Builder**

* Query Builder feature provides a simple and intuitive way to query the CMDB for CIs and relationships across multiple CMDB tables

### CMDB Dashboard

* It provides a single view of the quality of data at the CMDB, CI class, and CI levels using completeness, correctness, and compliance scores.

## **Terminology which we use during Implementation**

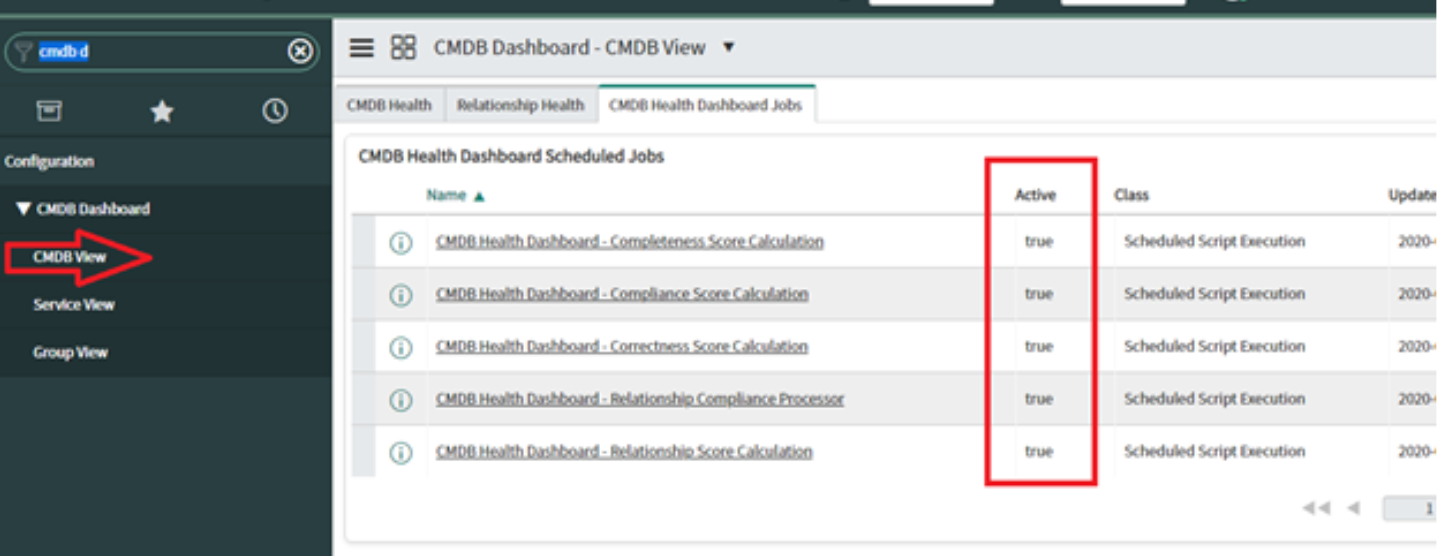
* We have to be familiar with some terminology, we will be talking about those all the time during CMDB implementation.
* **Configuration Item (CI):**A configuration item (CI) is any service component, infrastructure element, or other item that needs to be managed in order to ensure the successful delivery of services.
* **Attributes:** Information that further describes a CI such as a name, serial number, manufacturer, operating system etc.
* **Relationship:**Type: Relationship Types denotes the nature of relationship between a CI and another CI. It shows how the CIs are interconnected and interdependent with other CIs. For example, INSTALLED ON, Runs::Runs On.
* **Class:** It is a table that contains and represents a specific type or group of CIs that share common attributes such as a Windows Server, Linux Server, Printer, Virtual Machine, Vehicle, and Animal etc.
* **Base Table:** The core Configuration Item **(cmdb\_ci)** table, which stores the basic attributes of all the CIs. All configuration item classes extend from this table including all hardware and applications.
* **CI Classification:** CI classifications identify and characterize similar CI. Basically here we are classifying device type. If name start with “window” then it should go to **cmdb\_ci\_win\_server** table and so on.
* **CI Reclassification**: Ii is a CI whose class has been upgraded, downgraded, or switched. **For example,** a record upgraded from the **cmdb\_ci\_server** class to the Windows Server **cmdb\_ci\_win\_server** class.
* **CI Class Upgrade:** The CI class is updated to a class that is less generic in the class hierarchy, and the newly assigned class has additional attributes. **For example**, an upgrade occurs if a CI is moved from the Server **cmdb\_ci\_server** class to the Windows Server **cmdb\_ci\_win\_server** class. In another words we can say CI moves from parent to child class.
* **CI Class Downgrade**: Just Opposite to point 8.
* **CI Class Switch:** If you are moving CI which has different branch. For Example: Moving Linux server to windows server.
* **Parent and Child Class**: A table that extends another table is called a child class, and the table it extends is the parent class. A table can be both a parent and child class both extending and providing extensions for other tables. For example, the Server table extends the Computer table and the Windows Server table extends the Server table, thus making the Server table both a Parent and Child class.
* **Asset Vs CI:** On the Now platform, when creating a hardware asset, a corresponding CI will be automatically created or when a CI is discovered for the first time and inserted into the CMDB, a corresponding asset record will be automatically created.
  + **Asset:** Often starts during the procurement process, but may be created when a discovery tool finds the CI for the first time
    - Is part of the financial lifecycle.
  + **Configuration Item (CI):** Often starts when a discovery tool find the CI for the first time, but may be created during the procurement process.
    - Is part of the technical operations.
* **Asset DB Vs CMDB:**
  + Asset DB is inventory + financial data
  + CMDB is inventory + relationships
* **Application Service:** A set of interconnected applications and hosts which are configured to offer a service to the organization. Application services can be internal, like an organization email system or customer-facing, like an organization website.

## **CMDB Table and Required Attribute Details**

* **Hardware –** name, asset tag, installed date, status
* **Computer –** serial number, ip\_address, mac\_address, name, Model id
* **Network Adapter-**serial number, ip\_address, mac\_address, name, status, configuration item
* **Wireless Access Point**– serial number, name, Model id
* **Business Service**-Name, Business criticality, used for, operational status, service classification
* **IP Switch-**serial number, ip\_address, mac\_address, name, Model id
* **IP Router –**serial number, ip\_address, mac\_address, name, Model id
* **Infrastructure Service-** Name, version, operational status
* **Disk-** Name, Storage type, serial number, Model id
* **Application-** Name, Operational status
* **Software-**name, version, manufacturer

# **CMDB Implementation Best Practices**

* **Enable CMDB Health Dashboard:**By default, the CMDB Health Dashboard jobs are disabled. Need to enable it manually.
* Health indicators such as duplicate CIs, required CI fields, and audits contribute to the calculation of overall health of the CMDB.

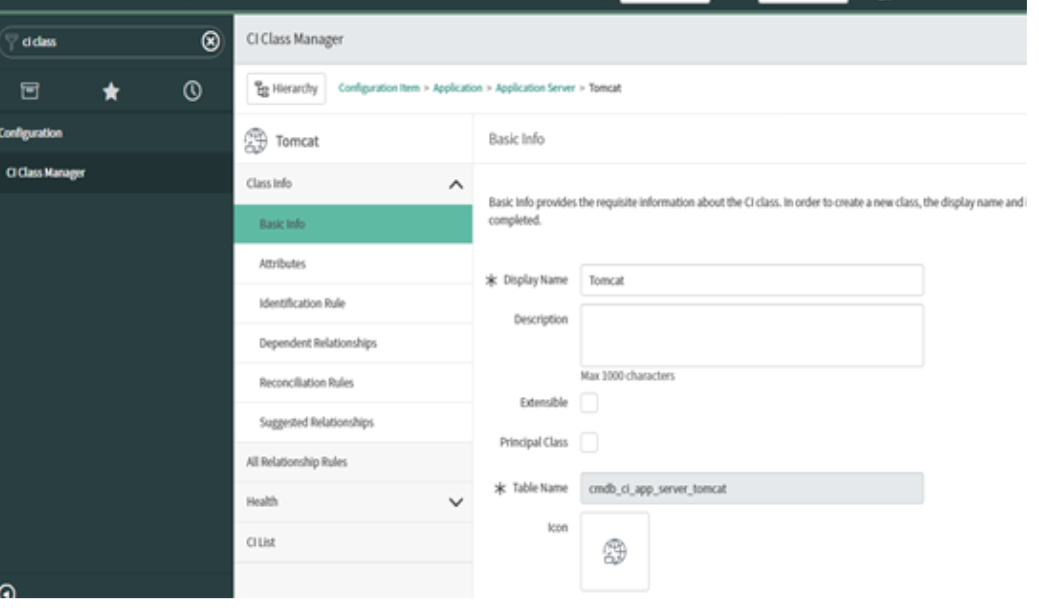


* **Utilize OOB CI Classes as much as possible:**Utilize CI Classes that are provided as baseline with Service Now.  It is also best practice to not extend off of CMDB\_CI.
* **CMDB attributes should be placed at the right level:**Suppose you want to create **Bios Date** attributes on the Windows, Linux and UNIX class instead of defining the attributes in the Server class. Best Practice: You should create **Bios Date** attributes in the Server class instead of creating three attribute on three different table.
* **Try to use a tree picker wherever it possible:**For example, you can use tree picker for location attribute. Rather than have the end user type in potentially duplicate or misspelled Locations use a tree picker.
* **Custom CMDB tables begin with “u\_cmdb\_ci”:**While creating new classes (Tables), make sure the table name starts with “u\_cmdb\_ci”.
* **You should not alter Relationships:**Relationships (in the table cmdb\_rel\_type) should remain baseline or out of box at all times. Changing them can break discovery and cause errors.
* **Avoid customizations:**Attributes should leverage the out of the box attributes rather than recreating with custom attributes.
* **Applications should have a Business Owner assigned:**Check to see that a Business Owner has been assigned to each Business Application.

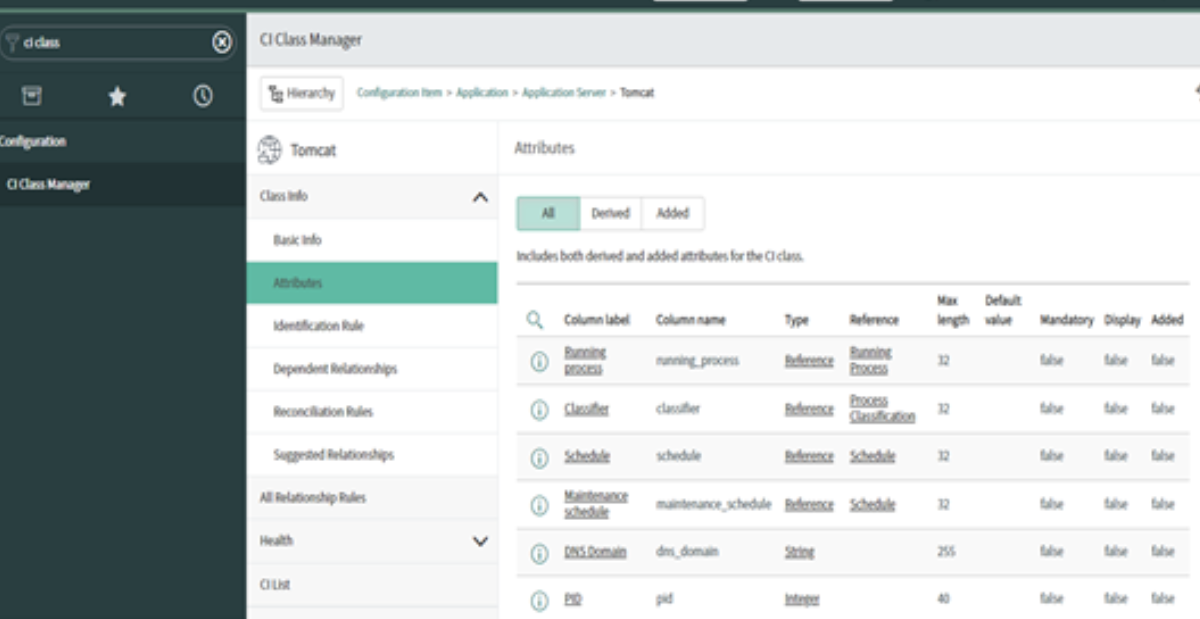
# **CMDB CI Class Manager**

## **Various sections and tabs of the CI Class Manager and their functionalities.**

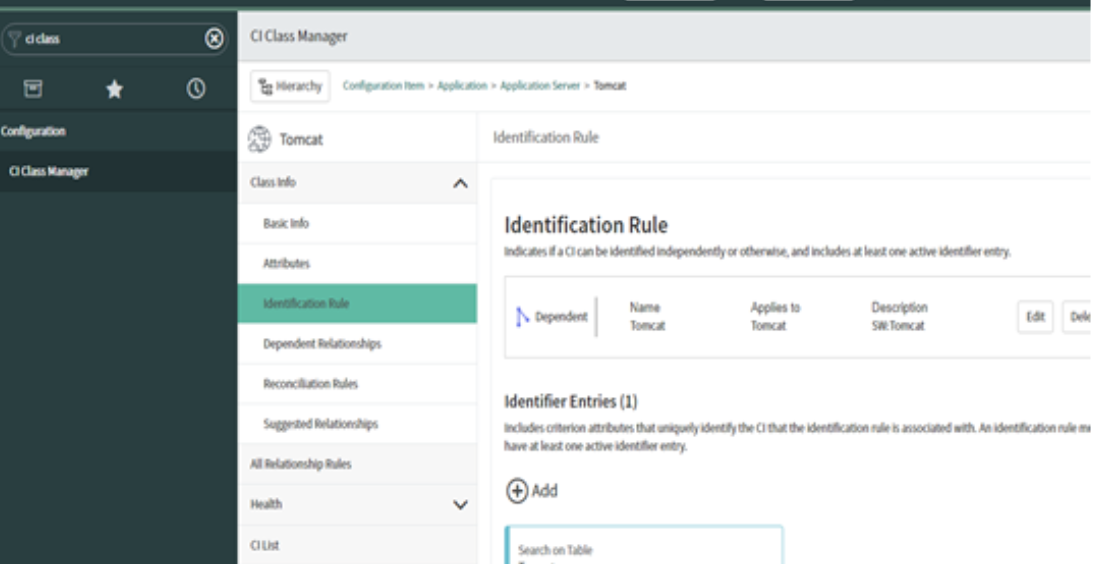
* **Basic Info**: The Basic Info tab provides a place to view or configure basic details such as a Display Name, Table Name, setting the Extensible flag, and associating an Icon to the class.



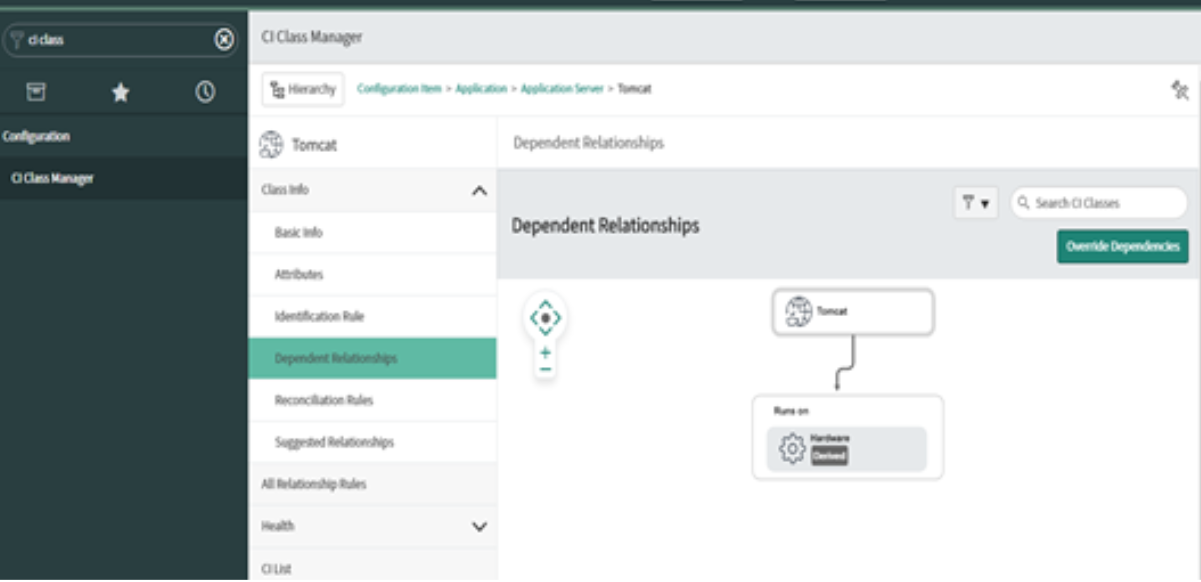
* **Attributes:**It allows you to view or to create new class attributes.



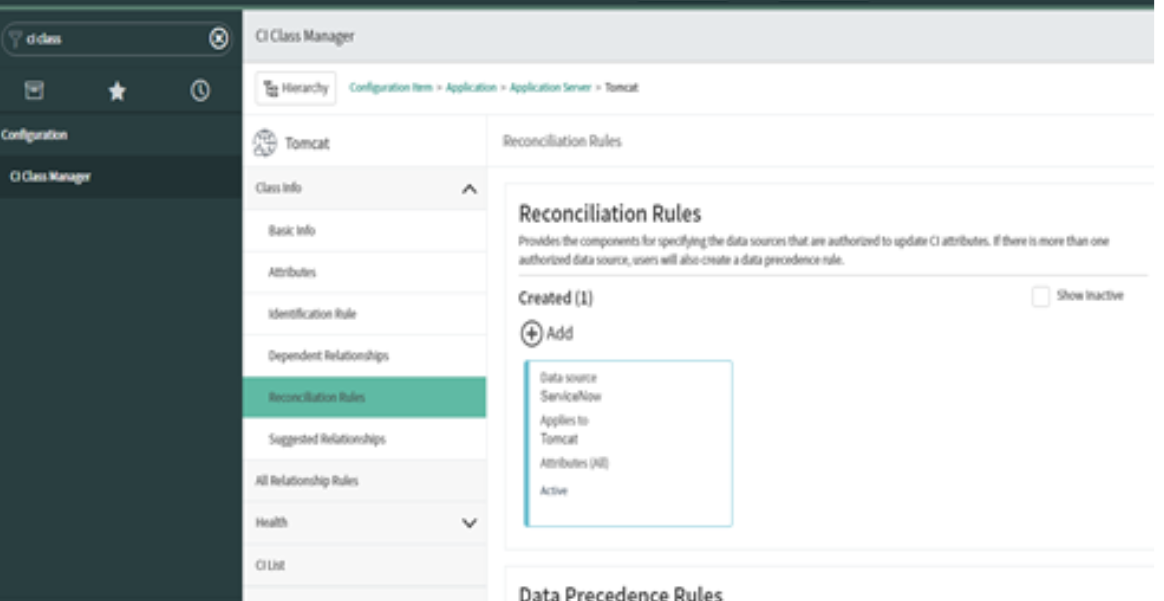
* **Identification Rule:**It used in the identification and reconciliation process to uniquely identify CIs. This tab by default displays the derived identification rule from the parent class as well as any defined identification entries.



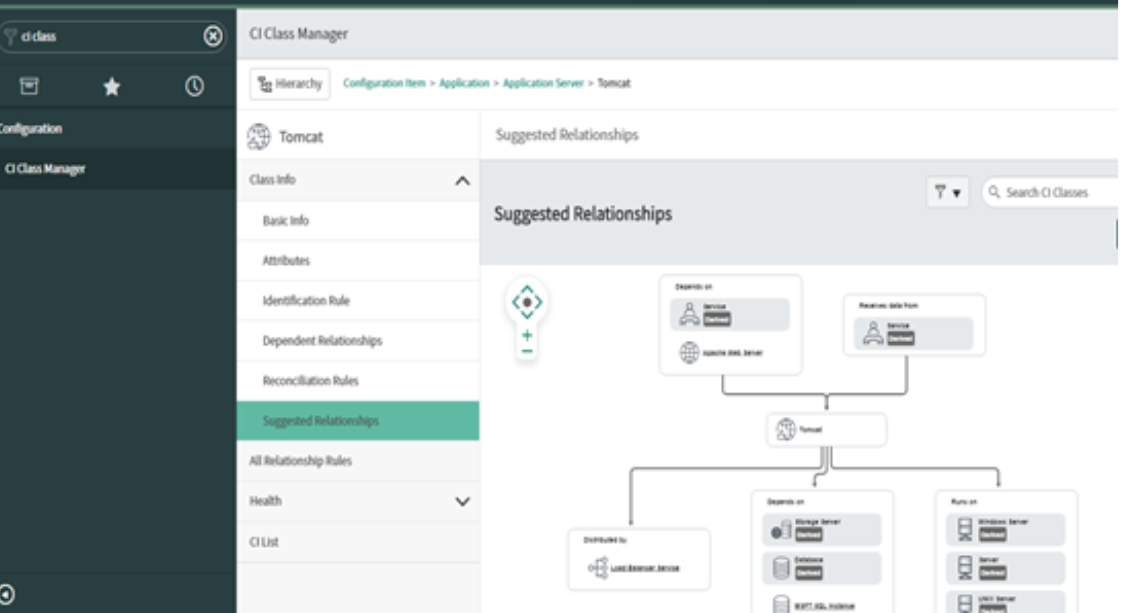
* **Dependent Relationship:** Create hosting and dependent relationship rules for CI classes to help with correctly identifying dependent CIs during the discovery and service mapping process.



* **Reconciliation Rule:** It is a process of reconciling CIs and CI attributes by allowing only designated authoritative data sources the ability to update the CMDB at the CI table and attribute level



* Provides the components for specifying the data sources that are authorized to update CI attributes. If there is more than one authorized data source, users will also create a data precedence rule.
* **Note:** reconciliation rules do not apply to the creation of records as any data source can create a CMDB record.
* Setting reconciliation rules prevents the flip flopping of various data sources updating each other’s values.
* **Data Precedence Rules:** Defines the priority across multiple data sources if more than one is authorized to edit the same CI attributes.
* **Data Refresh Rules:**Defines the frequency that authorized data sources should update CI attributes.
* **Suggested Relationship:**Suggested relationships are built to help the user decide which relationships are reasonable when building relationships for the class being defined.



* Take an Example:
* Allocated from: Allocated to
* Exports to: Imports from
* **Health:** Health metrics such as duplicate CIs, required CI fields, and audits contribute to the calculation of the overall health scorecards at the specific CI class or base CMDB level.
* The health of the CMDB data is monitored and reported for the following KPIs,
* **Completeness**: CIs are tested for required and recommended fields that are not populated.
* **Compliance:** The CMDB data is audited for adherence to pre-defined certificates.
* **Correctness:** CIs are tested against pre-defined data integrity rules such as identification rules (to detect duplicate CIs), orphan CI rules, and stale CI rules.
* **CI List:**The CI list displays all defined CIs for the selected class.
* **Pinned Class:** Pinning classes provide an easy way to navigate to the classes you use the most.  Clicking on the “Pin” displays the pinned classes.  Clicking on any of the displayed classes takes you to the settings for that class.
* **Summary of Automation Tools and Integrations**
* **Discovery Tools**: Network scanners (e.g., Nmap, Lansweeper) for automatic detection.
* **Integration Points**:
  + **CMDB**: Integration with tools like ServiceNow, BMC Helix, etc.
  + **ITSM**: Integration with change management and incident management tools to track and update asset records.
  + **Central Service**: Syncing with internal asset management tools for centralized visibility.