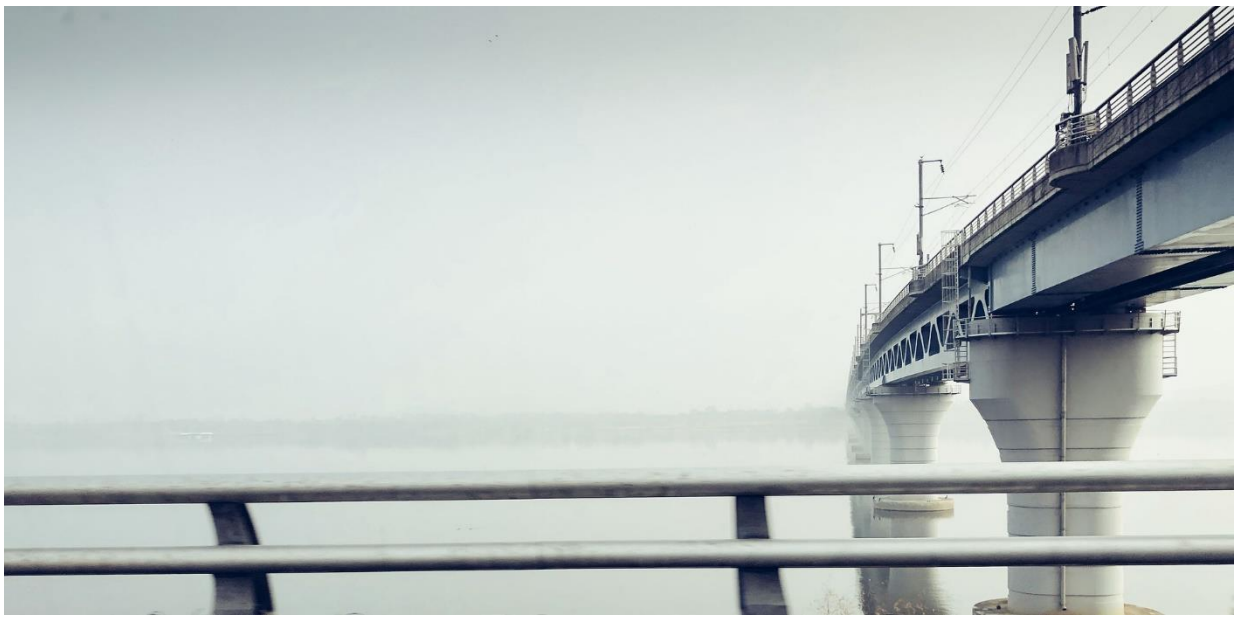
OBJECT GEARS

Guide to CMDB solution design



In this document we are describing how to begin with **Configuration management** process and with **Configuration Management Database** (CMDB), emphasizing clarity and practical examples. It is a general guide with some particular examples as they are solved in the Configuration database of ObjectGears platform.

Business context

Today when modern business is driven much by IT and IT becomes a business in many companies, **transparency of IT infrastructure** is a critical matter. As the infrastructure is growing and becoming more complex, we are facing challenges of consolidation, maintenance and ability to understand complex information about configuration.

Solution is **a configuration database** that stores information about IT infrastructure but not only about it – also about business environment / context and enables key functions of IT Service Management (ITSM) and other processes to use particular subjects of these processes – configuration items.

Practically every organization in each industry has to comply with regulatory requirements like Sarbanes–Oxley (SOX), Health Insurance Portability and Accountability (HIPAA) or Payment Card Industry Data Security Standard (PCI DSS). Other regulations are developed by particular governments or international organizations. One of goals of these regulations is to ensure system and sensitive data security, adequate management of these systems and data and proper ownership. Configuration management database is a fundamental tool for audit and compliance.

What is Configuration management

ITIL V3 defines Configuration management:

"The process responsible for ensuring that the assets required to deliver services are properly controlled, and that accurate and reliable information about those assets is available when and where it is needed. This information includes details of how the assets have been configured and the relationships between assets."

Configuration database is a prerequisite of a proper functioning of this process, a base on which the process can run. Configuration management is a subject of many other standards. In the bellow examples we will work with typical IT configuration items. However, in the same way we could solve also repository of used technologies to ensure clarity in products, services and their quality which is expected by their consumer/customer.

CMDB models your services and infrastructure. It is a system supporting proper decision making in IT, but it overlaps also to other areas including **audit, compliance or customer service**.

Scope of CMDB

Every organization solves the question of adequate scope of CMDB. Trying to cover all possible items and aspects will be costly, demanding on maintenance and time, cumbersome and therefore likely bound to fail. On the contrary, a model keeping little information will not reach necessary improvements. Therefore, we will show below how to set an optimal level of information detail. If we identify at the beginning fundamental entities which we need the most and we implement these, we are on a good track. Then we should create a plan according to which we will proceed.

Benefits and relationship to other processes

Institute of Configuration Management (icmhq.com) estimates in its study Monetary Value of Traditional Configuration Management versus CMII (published on January 14, 2014) that cost of sales can be reduced up to 40% by implementation of Configuration management due to minimization of various corrective actions that companies have to commonly apply today. If we looked for examples of these potential savings in IT, we would get these situations.

1. Every time when we are considering certain change we are meeting with administrators of the given systems and discussing relationships. To explain everything takes time repeatedly. (In case of Configuration database is implemented, we just display items that we want to see, and we can understand the relationships and context.)
2. When we are implementing certain change into an application, we often realize only in the course of works that originally passed information was not complete (there are other systems with which the application is connected and that are impacted by the change) or not accurate (e.g. another version of the application is used). Additional changes are costly, and we are repairing something which we already (incorrectly) did once. It costs us unnecessary money and we are losing time.
3. We want to renew a contract about hardware maintenance with a vendor. We have to go with colleagues to the server room to make sure which hardware we are actually using because we know that we cannot rely on our current records. Subsequently, we will be solving which hardware needs maintenance for the whole year and which just for a part of the year since migration to a new hardware is planned. We want to be sure that we pay an adequate support type (shorter response times by vendor in case of an issue mean higher fees for the service). However, we do not know which applications are running on all particular servers and how critical they are for business. In the end we pay support for all the servers for the whole year and rather a higher level of support because we are not able to trace everything. (In case of Configuration management process, we can filter servers for which the support is going to end. The necessary level of the support type can be derived from criticality of applications that are hosted by the server.)

Configuration management and Configuration database are related closely also to other processes. Below are some examples.

Change management

When we are planning certain change, it is useful to refer to particular items from the Configuration database and be ready to display other changes that are counting with it and display dependencies. We will avoid conflicting activities and can estimate well the whole impact based on the item relationship.

Release management

When we plan an application release, it is useful to mark items that are subject of the release both for a proper release approval and for ensuring that the release contains everything that should be deployed.

Incident and Problem management

If there is an outage or degradation of a certain service, we can easily identify items which were subject of a change recently and which may be likely a root cause of the outage. This will help to resolve the incident quickly.

If we find out that we have solved the same incident on the item repeatedly in the past (configuration item was repeatedly linked to the same incidents), it will be clear that we should pay attention to the root cause further, in the process called Problem management.

Asset management

Many configuration items are assets that we are managing also from the perspective of assets management. IT provides information about the status, location and changes to the Finance teams. This includes entering items in the inventory, regular inventory checks, questions of technical improvement and decommissioning.

Project management

Projects and project managers can get a valuable information from CMDB. It is suitable to determine already in the phase of feasibility study and project approval which configuration items will be impacted by the project and how. If we do so, we will significantly increase quality of this study and accuracy of cost estimates. We will also know which technical and business teams will be needed in the project. Without these considerations we are risking project delay and unexpected cost during project implementation. Project outputs should be also reflected in the Configuration database. Therefore, binding rules defining interaction of projects and Configuration management should be developed. Everyone in the organization should get familiar with these rules and follow them.

Information security

Configuration database provides an easy access to fundamental information for assessing information systems` security, risks and vulnerabilities. Information security team should also assess risks resulting from access rights of particular roles working in the Configuration database with particular configuration items. Sensitive data stored in the Configuration database might be e.g. IP addresses, patch levels, communication protocols, administrator names and their phone numbers.

How do we begin

If we are considering CMDB implementation we should map our current situation. You may keep records about applications, servers or hardware in Excel spreadsheets. If this is the case, you are on a good way. Although you might think that keeping these data in Excel manifests lack of professionality, important is that you recognize importance of these data, you have the data, even if not complete and with insufficient quality. However, this is something you can build on. You are already working on improving this process. Data can be easily uploaded from Excel to the Configuration database and then we can use CMDB reporting to identify key data that are missing.

If you do not have the data and the given information resides just in heads of particular colleagues, Excel is a good starting point. It will help us to clarify what we want to record for particular items and which

relationships they should have. At the same time, we will have to cope with cultural aspect of CMDB implementation.

Cultural aspect

When implementing configuration database, we will be solving the situation when we want to keep records of certain information. However, we will have to get them first from colleagues that are responsible for the given configuration items. Information represent power and therefore it might not be easy to get them. How to succeed?

Support from management and formal project approval

Approach **implementation of Configuration database as a project**: explain its benefits to management, get management support and establish a regular reporting about the project progress. Develop project initiation document where you will explain which current issues you want to solve, which goals you want to achieve and in which timeline, which resources (people, finance) you are going to need and why it pays off (business case describing both financial and non-financial benefits – e.g. minimization of risks given by lack of information about configuration base, solving audit finding etc.). If you come across issues hindering reaching the goals, e.g. in the area of missing data update, state this topic in the report of project progress as an *issue* including a solution proposal: application team manager will get a task to fill in data about missing application components and data exchange (interfaces). Support from management including a formal approval of your project, to which you can always refer, is very important. Like with any project you should have one (or max. two sponsors) – managers from the leadership team, who will stay behind you and who will support you and who have a real interest to make this project a success.

Buy-in from users

Every change gives rise to concerns or resistance. It is a natural human response. Dedicate enough time to explaining why this activity is launched. Prepare examples on which you can explain certain scenarios – current issues and how you want to solve them in future. You have utilized these for sure when convincing leadership team. Now you have to convince middle management and end users. There will be objections during these meetings for which you can have, but do not have to have, a clear answer immediately. It is absolutely ok to take a note of such a question or concern and come back to it later. Maybe you will not like listening to people saying that there were already many projects like this one or that the project is not worth of the effort and be sure that there will be such comments. In case of such reactions from some colleagues have a proper answer ready: refer to the leadership team decision to launch the project.

Successful CMDB implementation

Success of CMDB and Configuration management process implementation depends on the value added that the users will perceive. If people do not see benefits, they do not use project outputs and the whole initiative will fail. This should not be underestimated. It is true at the same time that CMDB

implementation represents a huge potential in most of medium-sized or large organizations. It is up to us to use this potential and make it a success.

Model development

If you decide to implement Configuration database, you will probably get it already with an existing model containing various types of configuration items, with various properties and relationships, something that the CMDB vendor describes as a *best practice*, at least within their experience.

It is important to note, that **there is not a single optimal model for configuration items**. There are various organizations of various industries and various size. Even for a particular company it is not possible to say that a certain model is optimal. In fact, it matters a lot who will be assessing the model, what is his or her experience with the topic, what did he or she solved already in the past. These personal perspectives lead to model adjustments which, however, are usually not essential. Much more important for the overall project success is identification of users with the given model, their buy-in, than more or less academic disputes whether some information should be recorded as a property or as a record in another class. The model should not be static, and we can reassess and modify it in future. Of course, such changes will require certain effort, but it is better to launch the project successfully than to stop it at the very beginning because people will not accept something that they find too complex.

Therefore, walk key people through the model, confirm that everything makes sense and make adjustments that will be necessary.

Example: It is useful to keep records of service covering for servers. Configuration item Server can be updated with a property Service contract (simple text field) which we will be able to fill in quickly. If we want to record data better, we can use a solution with several classes that will be more general and will contain more structured information See below figure.

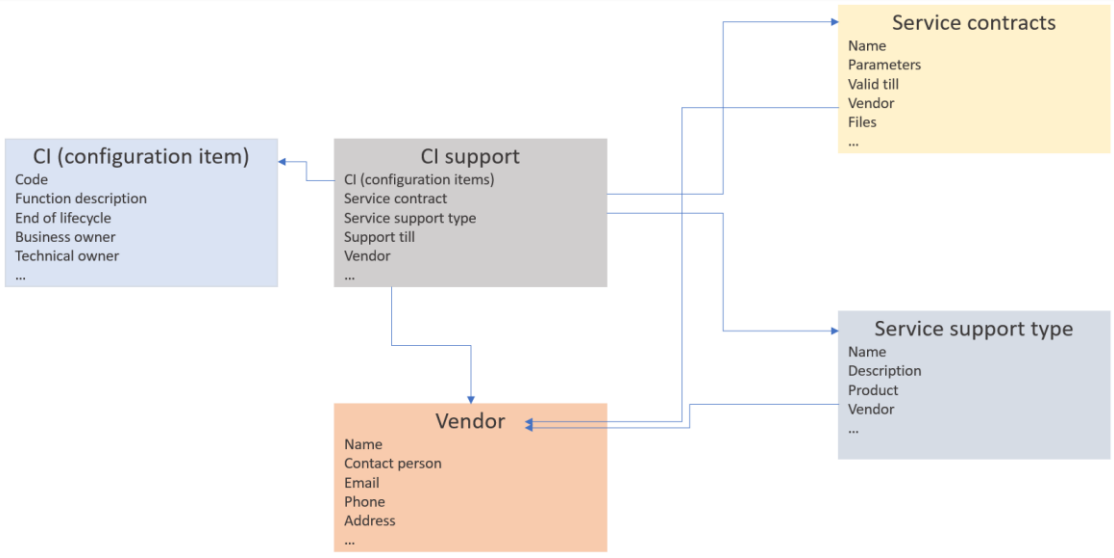


Figure 1: Service contracts enable us to follow necessary information about support of configuration items.

Items with which we begin

Configuration database can contain normally tens or hundreds of items. Following scheme represents an elementary distribution of the main configuration items.

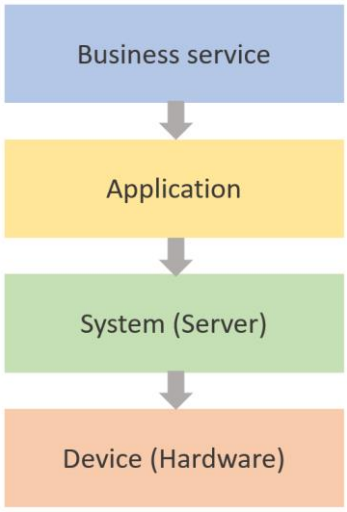


Figure 2: Elementary distribution of configuration items into layers.

Services represent securing of a particular business need, e.g. *Set up a new business partner/customer/vendor* or *Issue an order*.

Services are often provided by applications in which users perform activities for the service materialization.

Applications run on systems, servers with operating system.

These systems, servers, need a device (hardware), on which they can run. The model should be ready to capture also other scenarios provided by virtualization or cloud services.

It is useful to define items with which we will begin. It will help us achieve results and test the approach by means of which we will develop the model further. It is useful to start e.g. with the following entities: Devices (hardware), Machines (OS servers), Databases, Applications, Interfaces, Services.

Distribution of configuration items in layers

ObjectGears allows you to work with a model of your choice. The default model is distributing configuration items into the following major layers that may contain further sublayers.

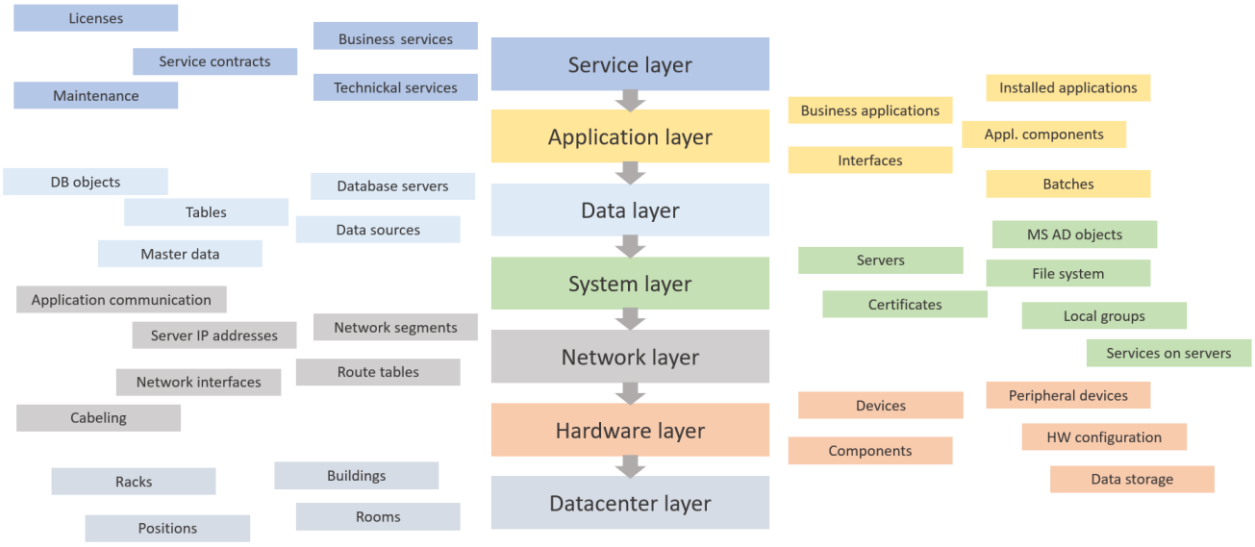


Figure 3: Detailed distribution of configuration items into layers.

At the initial implementation you will probably not use most of the configuration items. You can hide them and activate at the time when you begin using them.

Item identifier and item name

Unique and unchangeable name of configuration item?

Sometimes you can come across an opinion that each configuration item should have a unique name that should not be changes during its lifecycle. Actually, several things are mixed here. Below we will try to explain why this concept is erroneous and how to deal with this topic in a better, more practical way.

Item properties

Configuration items are represented by their properties that we decide to track in the Configuration database. These properties can be common to all the configuration items (e.g. unique item code that we will generate, description of function, end of the lifecycle or business owner that is approving configuration item changes and technical owner, i.e. administrator, that is supporting it and implementing the changes). Further properties can be specific for the given item type. Hardware devices will have some properties, database servers another and application and certificates again other properties.

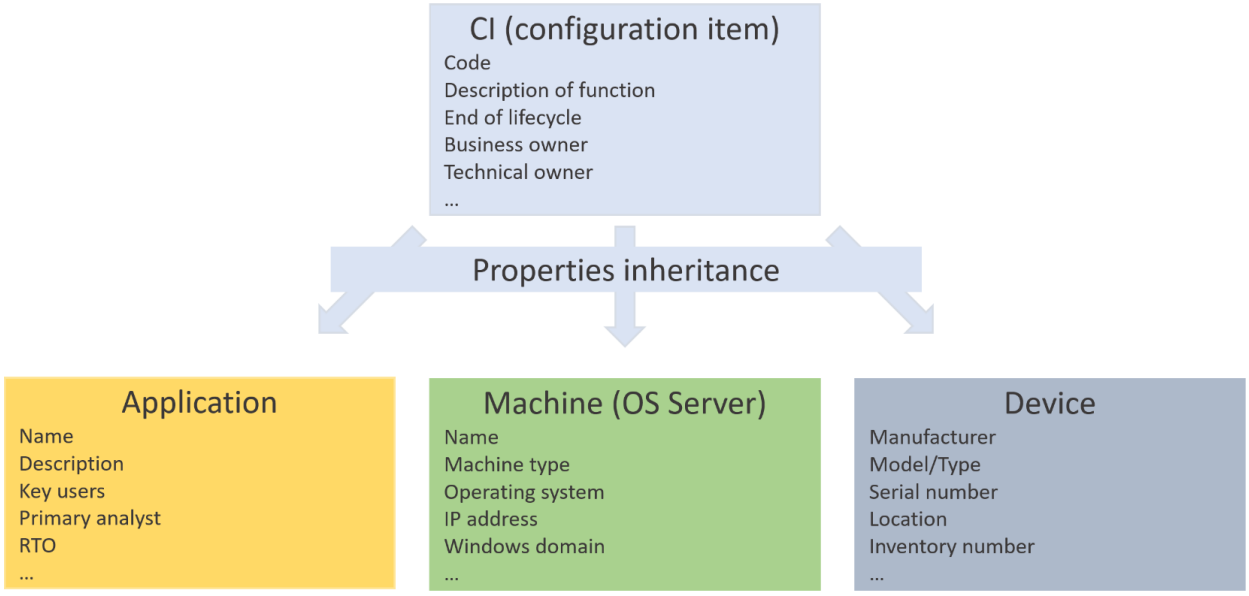


Figure 4: Particular configuration item types inherit common properties from the common parent class.

Identifying properties

We have to be able to distinguish between the items. Each type of configuration item should have certain properties, combination of which will be unique, i.e. we will recognize according to them which item we are dealing with. If we do not have such properties in the model yet, we should think how to improve our model. In case of devices such unique combination will be Manufacturer of the device, Device model and Serial number. In case of servers we may think about a single property: full domain name. However, if we consider that some servers can be outside of any domain or may exist in two separated networks having the same name, it is clear that we will need another identification. This can be IP address or network segment. However, these properties can change. Similarly, application name can be an identifier. Sometimes there is a good reason to change the application name and despite this change it will be still the same configuration item after the name change.

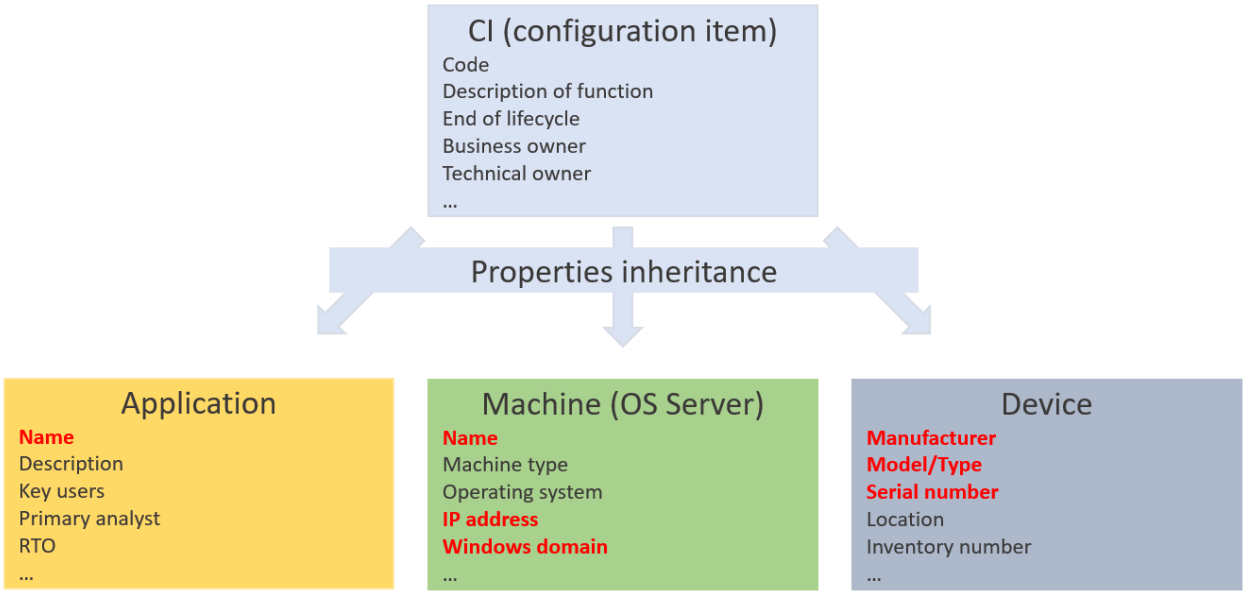


Figure 5: Identifying properties of configuration items are highlighted in red.

Unchangeable and unique identifier

If we are mentioning somewhere a configuration item, we should ensure that also after the change of certain properties, that occurs naturally, we will still know which item we are speaking about. If we refer to an item within the Configuration database this binding is ensured by a link (foreign key). If we are mentioning the item somewhere else, we should use the item code (e.g. CI00040970). Note that the code does not bear any information about name, manufacturer etc. We can derive from the code just the information that it is a configuration item (CI – Configuration Item). It is a unique code that is automatically assigned by the Configuration database to all the items. It is one of properties that are common to all the configuration items.

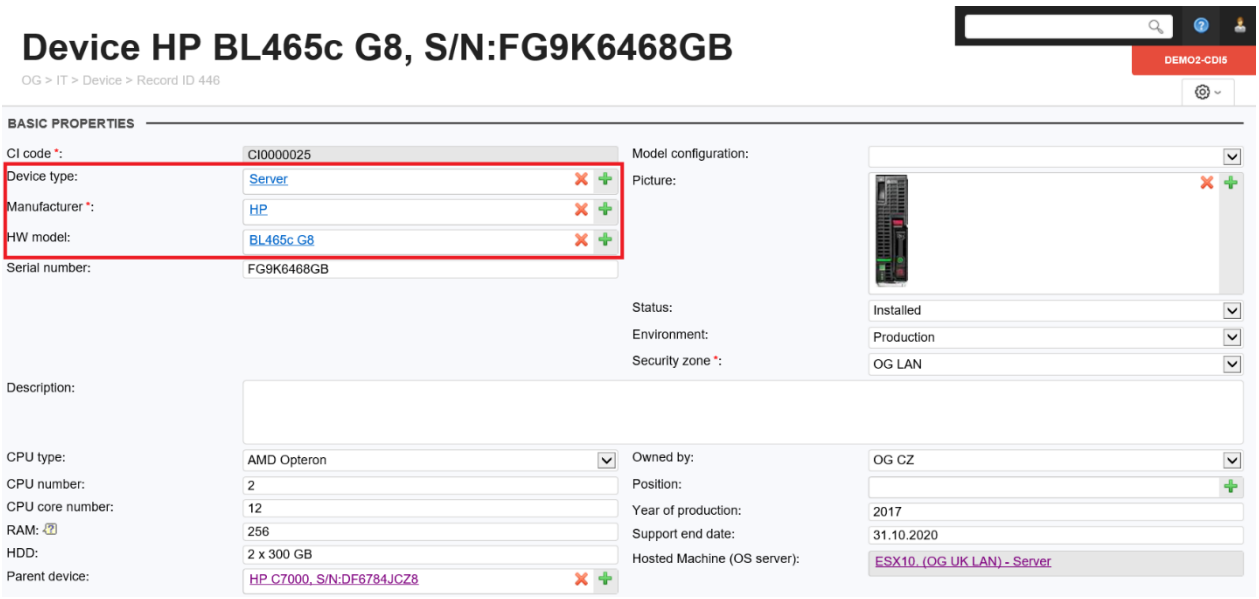


Figure 6: Example of a configuration item (device) – identifying columns are highlighted.

Item description

We will need also item description. It is an analogy of the name mentioned above but it has the advantage of being automatically built from properties that we will define for particular items. In case of application it can be Name and Environment (we should distinguish between the same application installed in Production and in the Test environment), in case of database it is Database server and name of the database, in case of hardware already mentioned Manufacturer, Model and Serial number. These are identifying properties. If we want to be sure which item, we are dealing with by reading the item description, these very properties will enable us that. There is one more difference compared to the above-mentioned concept of unique and unchangeable name – change of the description in time.

The description can change in time and it is ok

Imagine that you are migrating a database from one server to another within certain change. After migration it will be still the same database (the same configuration item) but one of its identifying properties (database server) will change. You will modify connection string of applications that work with the database and you will change the property Database server at the given item in the Configuration database. We would approach this in a similar way if we were changing the database name. Again, there is not a new item created in such a scenario and there might be a real need to change the database name. The description by which we are searching the item will be updated automatically. If we maintained the concept of unchangeable name of configuration item, such change would not be possible.

Type of item being part of the item description

It will be also useful to include type of the item into the item description. If we will be choosing from the list of configuration items, it will be clear to us immediately which items are we dealing with.

In the below figure you can an example from a Project record. The first added item is e.g. server DB-UK-01 which can be found in the network OG UK LAN. We can see application SAP in the modified items, then its particular instance used in Production and then the Interface between the production application instance of ObjectGears and production instance of Data warehouse (DWH).

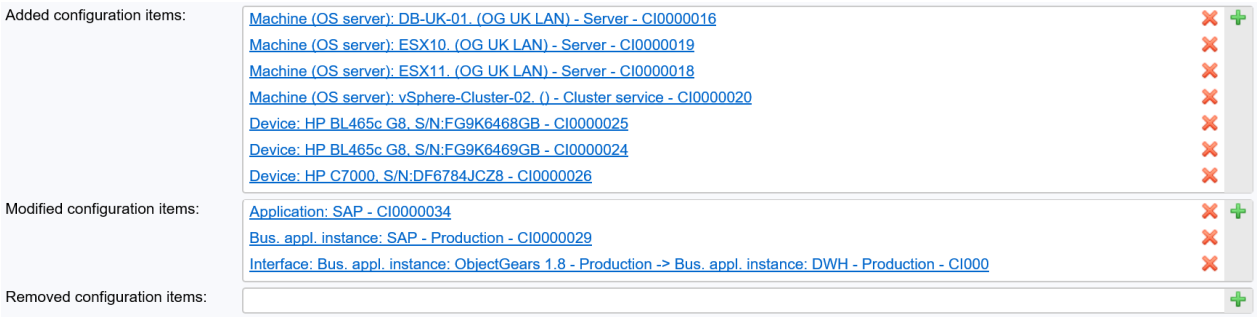


Figure 7: Part of the project card which is mapping project impact on configuration items.

Particular aspects of main types of configuration items

Services

Services represent securing particular need of business. We distinguish between Business services and Technical services. Examples of business services can be:

- Create a new business partner/customer/vendor
- Issue an order
- Check partner credit standing
- Invoice our delivery to customer
- Book an invoice
- Pay an invoice

Business services can be provided by technical services (see the concept of business and technical services in ITIL). From practical point of view, we recommend enabling also a link from business services directly to other configuration items. Practical experience shows that during implementation people want to express that a certain business service is secured by certain application and creation of technical services between business service and application seems to be complicated.

Applications

We can often see that certain application is too big unit if we want to express what is impacted in the application. E.g. a company using SAP may use one instance of this application across the whole organization. In such a situation it is useful to distinguish between particular modules or functional units. Besides configuration item SAP, we should create also other items for which we indicated item SAP in the property „Is module of ". In this way we can work with configuration items that we will call e.g. SAP FI, SAP MM etc. Similar approach will be useful for many ERP systems. If we will be solving an incident, we can better localize the part of the system that is impacted by selecting a particular module. Each module can have also different business owners.

Whereas with other entities (e.g. databases, device-hardware, server…) it is usually clear what is such an entity and what not, we have more difficult situation with applications. We recommend considering below criteria in such cases:

- Business considers the given system as an important application that is used in business processes
- In case of an outage it is probable that Service desk will be contacted
- Provides outputs with which the business works further
- Higher number of users
- Requires licensing

Besides properties of applications we will also analyze relationships to other entities. We recommend to use IT model that is part of the ObjectGears installation as a starting point and adjust it according to your needs. For applications we should consider following relationships with other entities.

- Interfaces – this entity is defined by an application that is passing the data and by an application that is receiving the data.
- Data sources – databases and other data sources used by the application
- Infrastructure – the link with infrastructure is realized by application component which is running on certain server
- Processes – application supports certain process
- Key users (Job positions)
- Entity Change and Release management, Test and Project management, Incident and Problem management, Business and Technical services that work with application like with other configuration items

Interfaces

Interfaces represent a crucial relationship between two configuration items – applications. We are keeping records of applications providing the data and applications consuming the data. Besides description of the reason for data interchange we should track also frequency of the data interchange, data entity that is transferred, particular interface and interface technology.

Data sources

Data in databases can be shared with various applications. It is useful to track these dependencies. We should track name of the database, database server that manages it, collation, compatibility level, date of the last back-up, size, available space. Besides classic databases located on database servers we can come across also other data sources. Business is sometimes using so called **End User Computing** applications. These are e.g. applications based on MS Access or MS Excel. We may work also with other applications having their own proprietary databases. In case that these applications support critical processes, we should capture also their data sources. If we start keeping track of the current state in our organization, we are making the first step to improvement in the area of inadequate applications. Mapping and naming are the first steps to remediation.

Servers

It is useful to distinguish between servers (systems with operating system) and devices (hardware). Devices are physical boxes installed to racks or standing outside of racks. Servers are installed on devices. Key server property is operating system, e.g. Windows Server 2016, Windows Server 2019, Linux etc. Both entities have different properties.

Servers can be **physical (native)**, that are installed on hardware, or **virtual**, running on a virtualization software (e.g. VMware or Hyper-V). Furthermore, we should be able to capture **clusters**, consisting from nodes of physical servers, **cluster services** running on clusters or **farms** bundling several physical or virtual servers together. We are probably going to track much more other properties for servers – DNS name, whether the server uses DHCP reservation, IP address (in primary and secondary locality), maintenance window, server name, patching category, to which the server belongs, security zone, in which the server is contained, number of CPU, RAM size, size of disks, Windows domain, preferred, current and failover node …

Device (Hardware)

For devices we want to know manufacturer, model, serial number, physical location, security zone, number of CPU, their type and frequency, size of RAM, size of physical disks, owner, inventory number, parent device (e.g. blade enclosure for blade servers), year of production, end of support, service contract…

Automated vs. manual data update

Many properties do not have to be filled in manually. We can read them from systems by means of infrastructure scans. Such properties shall be set to read-only because they will be updated automatically. Examples of these data are devices (hardware), servers, databases or certificates. We can detect also software installed on computers. Most of business applications that we are using, however, cannot be scanned. These can be modules of larger systems or on the contrary small applications that are not installed. There is a similar situation with interfaces. It is possible to analyze the communication of applications, but we will usually get more clear information when discussing with application administrators.

Do not hesitate to use systems of third parties as data sources. Configuration database software can indeed often read data directly from various systems (see e.g. ObjectGears infrastructure solutions), however, this ability should not be the major criterion when deciding about a particular solution for Configuration database. There are many specialized tools these days that can get data from various systems and store them in form of files that you can upload into CMDB. Many systems can be managed today (and information read from them) by means of PowerShell. These are not only systems of Microsoft but also of other vendors that implement **MMC (Microsoft Management Console) snap-ins**. Much more important ability of CMDB is to model, visualize and use the data in other processes.

Baseline

In certain cases we want to check whether items keep certain (approved) properties or whether a deviation occurred at these properties. We can keep control of critical data at each item in this way and after scanning the actual status we can compare approved and actual values. We can be alerted about deviations when displaying the given item, by an automated email notification or when displaying a corresponding report. You can always easily change the baseline if you want to declare actual status as an approved one and not to be alerted about the deviations anymore.

Propagation of manually set values into systems

Data in the Configuration database can be not only collected from the target systems, but you can also set target systems according to the values entered into the Configuration database. Modern systems can be administered not only by graphic user interface but also through their program interface. So, you can e.g. create virtual servers on click according to parameters that you have chosen in the Configuration database.

Data visualization

We should be able to visualize data in our configuration database. Schemes enable us to focus our attention to particular relationships and quickly understand necessary context.

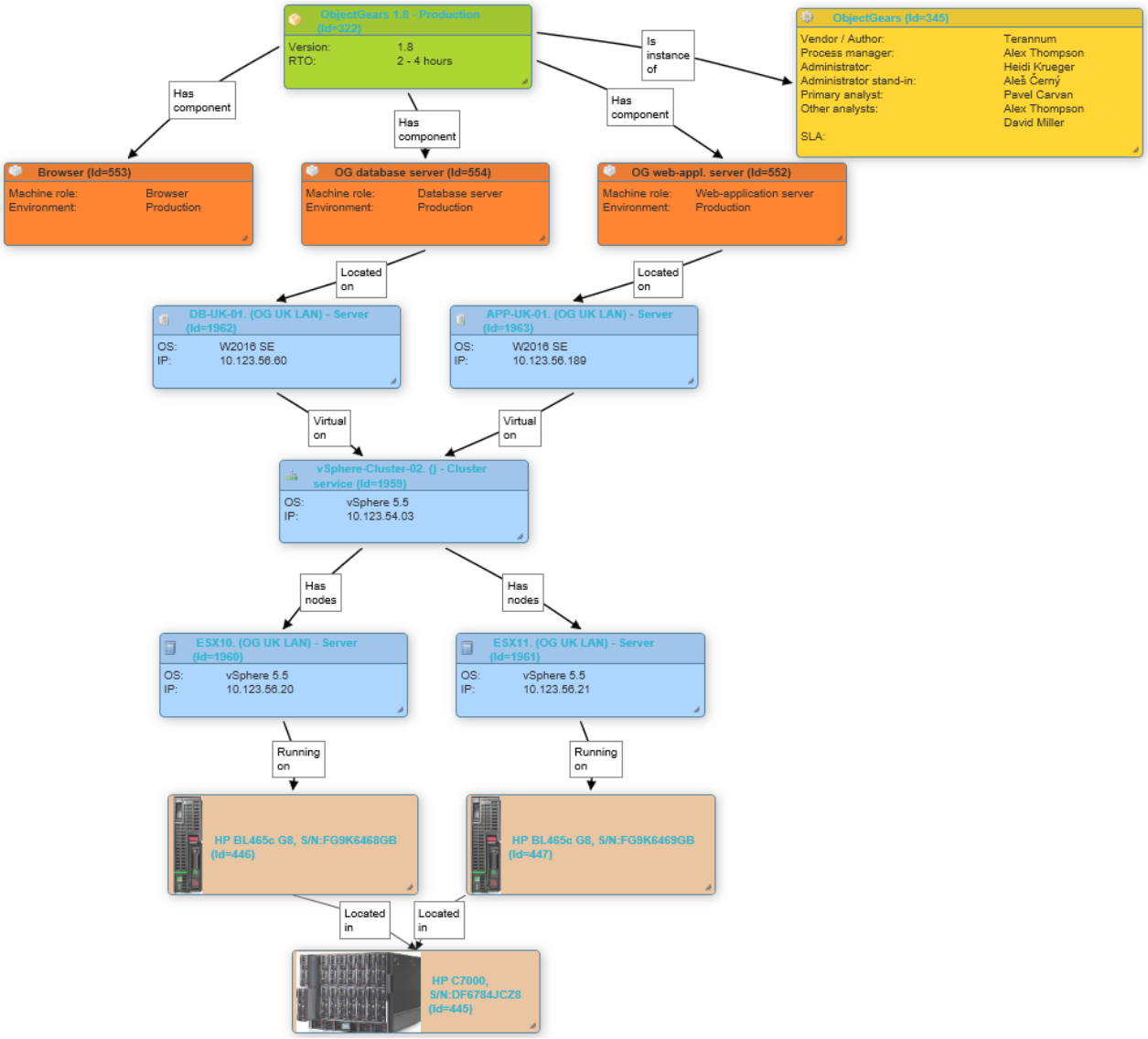


Figure 8: Scheme representing relationships of configuration items.

Missing critical data

We should define which data are crucial (obligatory) for each entity and we should regularly check percentage of cases when they are filled in. Missing data should be provided by the owner of the given configuration item.

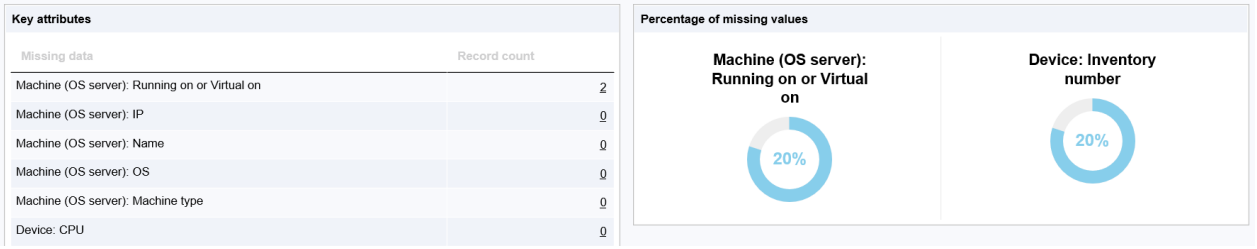


Figure 9: Missing data in key properties – number and their relative share, drill-down to concerned records.

Summary

This guide is providing a basic framework for configuration database design. Below you can find summary of the key concepts:

- Configuration database models services, applications and underlying infrastructure (data sources, servers, hardware etc.).
- Successful implementation of configuration database and configuration management processes depends on ability to demonstrate the value added.
- Cultural aspects have to be reflected during implementation in order this system is adopted in a positive way.
- Implementation and development should be planned. Relying on an organic development without a regular plan revision does not lead to optimal results.
- Establish a steering committee that will regularly deal with issues that emerged and prioritize CMDB development.
- Adapt the default model of the Configuration database according to your needs.
- Define *Short description* for each class of configuration items as a combination of identifying properties of these items. This description will be dynamic and will reflect change of identifying properties during their lifecycle. Use the sequence that is generated by CMDB as an unchangeable identifier.
- Pay attention to what you consider as an application. Certain configuration items have pretty good definition of what should be considered as these items. For applications we usually do not have a trivial definition and more aspects have to be considered.
- Define properties and relationships between configuration items. Monitor critical properties and relationships in overviews of missing data. Agree on responsibilities: who will add which data and till when. Read the data from infrastructure by automated scans and compare actual data with those intended/approved, where appropriate.
- Link other processes to the configuration management (Incident and Problem management, Change and Release management, Project management). By this you will clarify dependencies and when working on tasks or planning changes you will know the configuration item history and context.

Contact:

Pavel Carvan Author focuses on Configuration management topics for a long period of time.	Email: pavel.carvan@objectgears.cz LinkedIn: https://www.linkedin.com/in/pavel-carvan-96676835/