ITIL v3 and your CMDB: using actionable federation to create a configuration management system

White paper

Table of contents

Introduction

IT initiatives such as IT-business alignment, service management, data center transformation and application quality are driving IT organizations to pursue greater information transparency across management domains, using the perspective of services provided to the business as a common context. The IT Infrastructure Library (ITIL) Version 3 provides a practical response to this need with the concept of a configuration management system (CMS).

A CMS creates a common view of business services by providing access to service information across specialized IT management silos, thus facilitating an IT organization's transformation from a focus on technology to a focus on business outcomes driven by business services. Rather than creating and maintaining a single monolithic configuration management database (CMDB) that physically stores all service information, a federated CMS features an integrated CMDB that cannot only store configuration items (CIs) and model their service relationships, but also dynamically access other data sources to provide all IT management domains with a more complete, common understanding of business services. Thus the integrated CMDB provides a single point of access to multiple federated data sources without becoming the single repository. By using a federated approach, IT organizations can quickly access current information across teams and tools in a common service context, resulting in faster, better, business-aware decisions that improve business service quality and reduce cost.

In a federated CMS, the integrated CMDB uses a set of federation services that facilitate queries between clients (users and applications that request service information from the CMS) and external, autonomous data sources that maintain additional information about the configuration and management of services. Federation services are tightly coupled with the service data model in the integrated CMDB so they can dynamically associate data in external data sources (extended service data) to the CIs physically stored in the integrated CMDB (core service data). Thus, the integrated CMDB provides CMS clients with a seamless, single point of access to service information that may be distributed across multiple otherwise-incompatible systems.

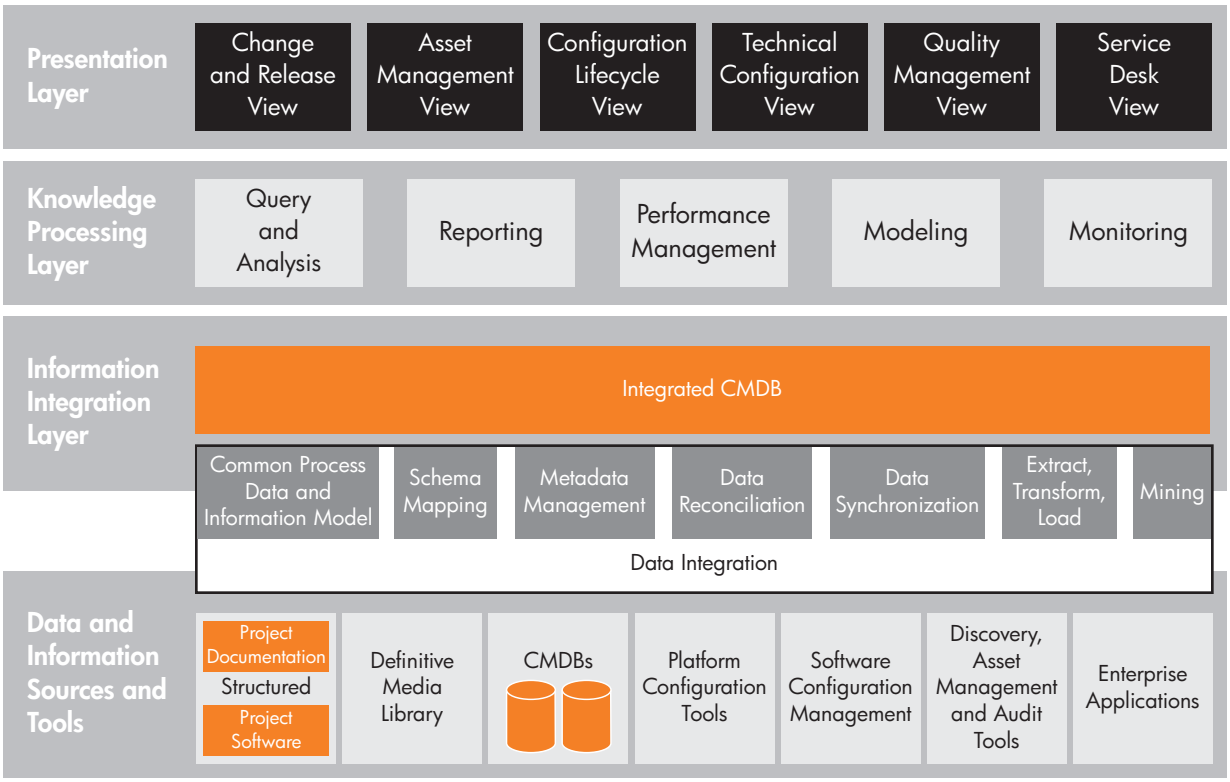
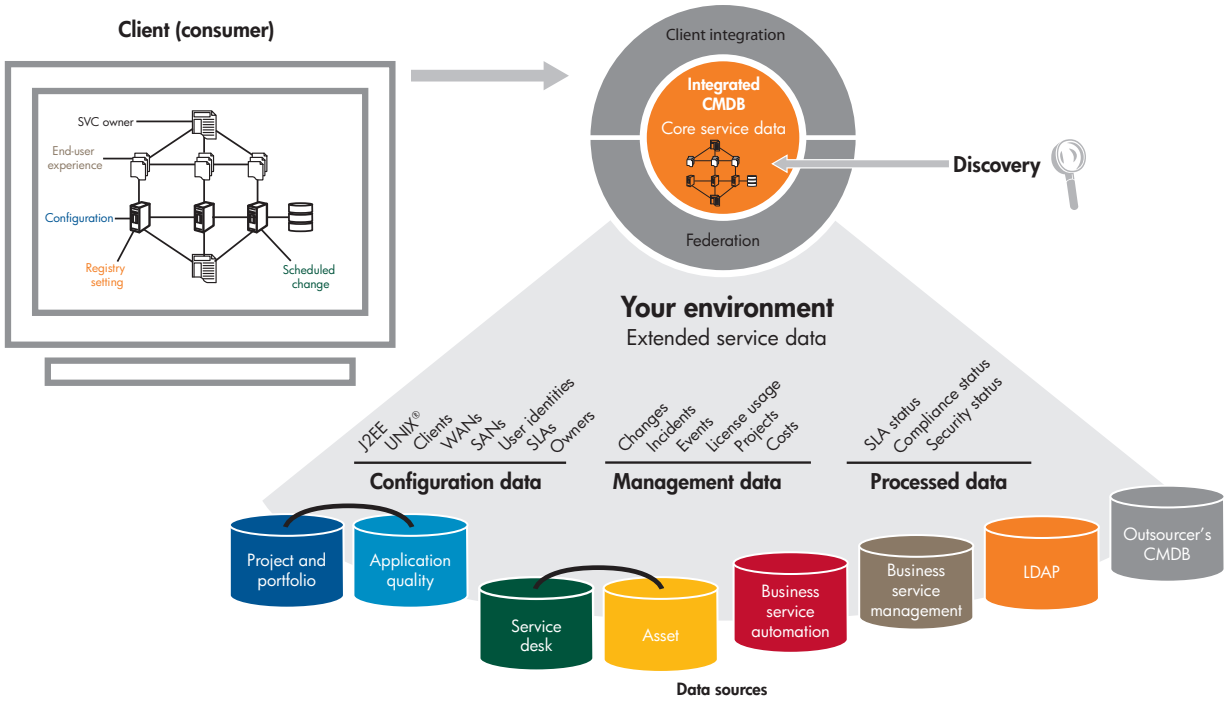


Figure 1. Whereas the CMDB is characterized as a database, a configuration management system includes tools for information integration, knowledge processing and presentation. The integrated CMDB provides a unified point of access to information from a variety of sources.

Figure 2. With actionable federation, rich service data from a variety of sources is delivered within the user interface and data model context of client applications. Client applications and users do not need to know where the external information resides.



What is actionable federation?

A key requirement for increasing value from a federated CMS is verifying that the information it provides is actionable, i.e., usable by applications and users to make decisions and take action that improves business outcomes. With an actionable federation approach, federated information is accessed and used just like the application's own native data. The key to actionable federation is a combination of transparency and context.

Behind the scenes, queries from client applications involving CMS data are directed to the integrated CMDB, which in turn queries both the physically stored data in the integrated CMDB and federated data in external sources. The physical location of the federated data is transparent to the client application. Query results are then delivered to the client application in context of its own data model and user interface so they can be programmatically consumed, displayed, sorted and acted upon by the client application's business logic.

Thus, actionable federation allows IT specialists from one domain (e.g., network management) to use their familiar tools to access information from data sources in other IT domains (e.g., application management) without learning how to use other interfaces. As far as they can tell, the application they work with every day now simply has a richer set of information available to it.

Key requirements for actionable federation

Actionable federation requires CMS federation services that work together to provide both common service context and coordination of information access between clients and federated data sources. To better understand requirements for how CMS federation services work together, let's look at a travel website analogy.

Arranging a trip has been made far simpler and more powerful through travel websites such as Expedia, Travelocity and Orbitz. Any one of these sites provides a single logical information source that federates across many disparate sources, from airline, hotel and rental car systems to street maps, attractions and activities. A single query to the travel website returns a coordinated set of answers spanning air, car, hotel and even sightseeing attractions. These results can be sorted, reworked based on new parameters such as different travel dates, and ultimately transacted upon.

Just as the travel website provides a single point of query about virtually any aspect of your trip, CMS federation does the same for business services, from infrastructure components to applications, detailed configurations to specific management data, regardless of where and how that information is independently managed and stored. Now let's focus on some of the underlying capabilities that make it work.

The close interplay between the service data metamodel and the integrated CMDB data model is one reason why federation services and an authoritative integrated CMDB should be tightly coupled: it creates a unified model for understanding core and extended service information.

- **Common service context.** The travel site can make sense of different information from different vendors by relating it to the concept of a "trip." It understands that a trip can have multiple components including flights, rental cars, hotels and attractions. It understands that flights have certain attributes such as duration (measured in hours and minutes) and fare classes. And it knows important ways in which information about one trip component relates with another, so it can coordinate your car pickup time with your flight arrival time, or find a hotel close to a landmark.
- **Data source registry.** The travel site doesn't store data about all the flights, car rentals and hotels available, but it knows where to look. It has a registry of information needed to extract data from data sources for airlines, car rental companies, hotels and more.
- **Common reference points.** To find a flight and a hotel for your desired destination, the travel site uses common reference points such as cities and street addresses to look up flights, hotels, cars and attractions in external databases managed by airlines, hotels and cars. What the user enters may not be exactly what these external databases use, so reconciliation between reference points is needed. For example, if the user's query specifies "San Francisco," the travel site will need to reconcile to airline systems' records of flights with airport code = "SFO." Or if a zip code is provided, it may reconcile to hotel systems' properties with addresses containing the nearest three zip codes.
- **Query.** Using its data model for a "trip" service, the travel site helps a user compose a query with valid structure and parameters (e.g., travel dates based on a web calendar); parses and translates that query into different queries understood by all external airline, car and hotel systems; submits the queries to each data source; and then gathers the results and returns them as a single answer to the user.
- **Translation.** If the travel-site user specifies a hotel room with "high-speed Internet access," the travel site may need to translate this term between different hotel systems' representation of this amenity. Perhaps it may need to query Hilton for "Internet = yes" and Marriott for "in-room Internet" = yes. No matter how hotel systems format their data, translation allows the user to use a single term, and the travel site to display all the results in a common apples-to-apples format, so it is easy for the user to understand and act on.

The analogy above highlights the fundamental importance of five actionable federation requirements.

Requirement #1: service data metamodel

Just as a travel site maintains a model for information about a "trip" as its common service context across the many systems it communicates with, an integrated CMDB needs a common model of information about a business service that spans its physically stored data and federated data sources. With actionable federation, the federation service that creates and maintains models of both core (physically stored) and extended (federated) service data is referred to as a "service data metamodel."

A service data metamodel doesn't hold any data itself. Rather, it describes the structure and format of both core service data stored in the integrated CMDB plus extended service data available in federated data sources—and how they relate to each other. In the case of core service data, the service data metamodel determines the actual data model instance in the integrated CMDB; that is, it sets up how the integrated CMDB will accept, store, manage and understand CI data. For extended service data, it determines how data that is dynamically accessed from external repositories will relate to CI data stored in the integrated

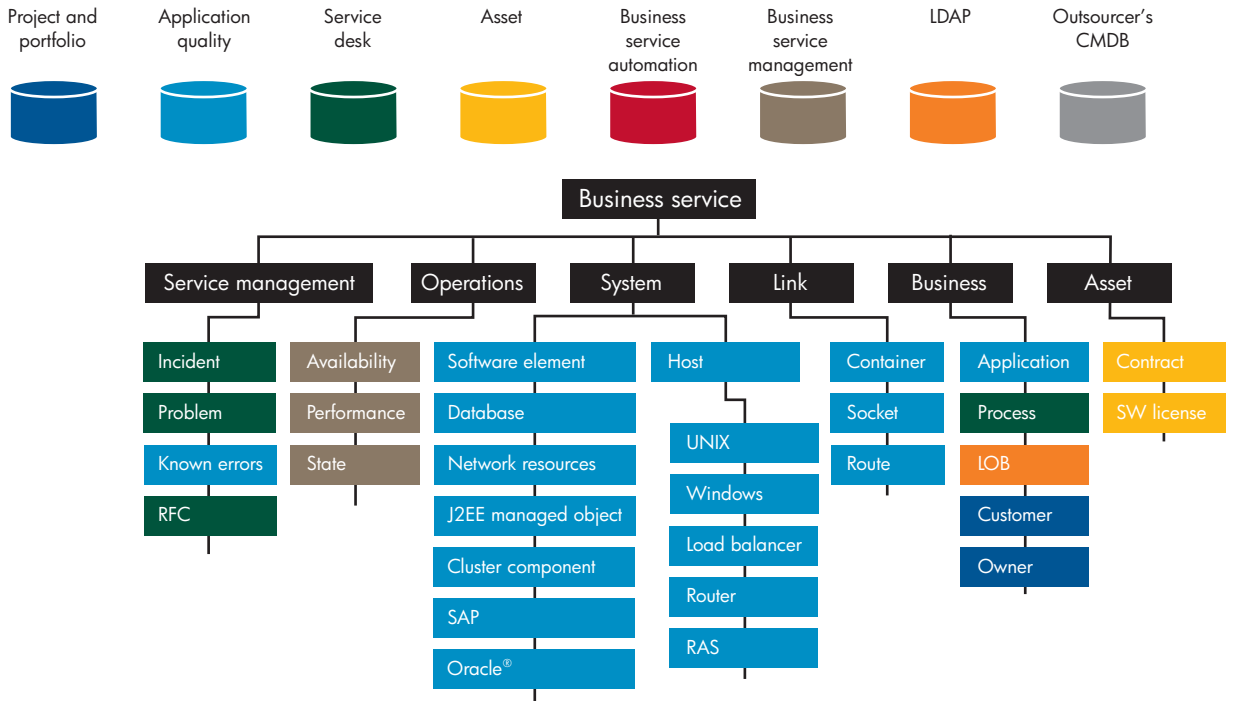


Figure 3. The service data metamodel maintains data relationships between CI data physically stored in the integrated CMDB and extended data in federated data sources.

CMDB. For example, the service data metamodel may determine that performance and availability data may relate to infrastructure CIs, but not people CIs. This close interplay between the service data metamodel and integrated CMDB data model is one reason why federation services and the integrated CMDB should be tightly coupled: It creates a single model for understanding core and extended service information.

Of course, a service data metamodel cannot start out as a comprehensive universal data model for all of IT management. But it is a central place where new management data types can be incorporated as they need to be shared. As process maturity and data sharing among IT management domains expands, and more data is selected for cross-domain access, the service data metamodel expands with it, creating a valuable, strategic asset in an IT organization's ability to work across teams to improve service quality and reduce cost and risk.

By using interrelated CIs stored in the integrated CMDB as the reference points for federation, users and management applications can quickly find data related not just to a single CI, but also related CIs, from a business service to an application and down to a network port. For example, if a change management application user wants to understand the impact of a change to an SAP Financials application server, she can issue a single query about "SAP Financial application server and related CIs" and have the integrated CMDB do the work of identifying which related CIs to include in queries to federated data sources.

Getting this core service information right in the integrated CMDB is fundamental to an effective CMS, and why dependency mapping is worth special consideration.

Dependency mapping

One reason behind many of the failures of early CMDB efforts was the lack of mature technology to create and maintain service dependency maps and a reliance on highly manual methods of CMDB population. Out-of-date, inaccurate CI information lacking the context of relationships to applications and services often led to a lack of confidence in the CMDB project and eventual failure to adopt. To properly plan for and execute software changes, for example, the CMDB should be able to render details such as database tablespace settings and J2EE sockets of custom applications, and understand the impact relationships between components.

Application dependency mapping software is a powerful class of auto-discovery tools that probe the IT environment not only for software and devices, but also the intricate details of their relationships to enable a business service. The most important considerations for application dependency mapping is the level of accuracy and detail provided to the integrated CMDB, and the ability of the tool to dynamically map what it finds to applications and services.

To maintain an accurate and useful model of the IT environment, several capabilities are important. First, the tool should provide discovery users with fine-grained control over the discovery logic to discover custom and legacy applications as well as more accurately discover packaged applications. Second, automated rules for maintaining mappings are important to verifying that the integrated CMDB's service models of discovered infrastructure and applications are accurate and efficiently maintained.

Finally, more powerful auto-discovery solutions also provide tools for defining impact rules that specify what the impact to an application or service would be if a particular CI were to suffer in performance or availability. This information can be used by change management to automate what-if analysis for change risk assessment, by operations to determine the business severity of performance and availability events, and by the service desk to determine incident priority.

Requirement #2: data source registry

Just as a travel website needs a data source registry to know where and how to access the latest flight or hotel data, a CMS needs to know where and how to access different types of extended service data (outside the CMDB). With actionable federation, the federation service that creates and maintains information on where and how to access extended service data in federated data sources is the data source registry.

A data source registry determines which federated data source will serve as a source of truth for which kinds of data by mapping which classes of CIs and management data in the service data metamodel are owned by which federated data sources. For example, the data source registry would know that router performance status is provided by a particular network management application. The data source registry also contains information on how to access data from the appropriate federated data source, such as the path to a federation adapter.

Thus the data source registry, working in conjunction with the service data metamodel, provides transparency: the client system or user does not need to know what system holds what kinds of data.

Requirement #3: CI identity reconciliation

Just as a travel website needs common reference points such as cities and addresses to match up a user's query to information in flight and hotel systems, a CMS needs to match up a user's query about CIs with extended service data related to that CI in federated data sources. With actionable federation, the federation service that identifies one or more matches between a CI in a query and a representation of that CI in a federated data source is called CI identity reconciliation.

Even when multiple systems (e.g., asset management, event management and server automation) have a notion of the same CI in the physical world (e.g., a server), they often don't identify that CI in the same manner. In our travel website example, we saw that a user may specify "San Francisco" while an airline reservation system uses the airport code "SFO." In a CMS, one system may refer to a server using a serial number, another a media access control (MAC) address, and another a host name. CI identity reconciliation is the process of matching up the client system's representation of a CI with a federated data source's different representation of that same CI.

For example, if a service desk application wants the real-time operational status of what it knows as "blade server XYZ," but the event management system that monitors operational status knows the same system as "blade host 123," the query about "blade server XYZ" from the client application must be reconciled to "blade host 123" in the federated data source in order to obtain the operational status of the correct device.

Requirement #4: dynamic query

In our travel example, we saw how the travel site helped the user compose a single query with a single result from multiple data sources. Behind the scenes, the initial query was automatically parsed, translated, submitted to the appropriate data sources and compiled into a single result. Likewise, in order for federated data to become actionable information in a CMS, the information should be transparent and in context of the action or decision being considered by the client. In an actionable federation approach, both transparency and context are achieved through dynamic query.

A key concept for actionable federation is that the integrated CMDB provides a single point of query to multiple federated data sources without becoming the single repository. Federation services are tightly coupled with the integrated CMDB so it can associate data in federated data sources to the CIs represented in the CMDB, creating a virtual data model combining the data in the CMDB and the federated data sources.

With actionable federation, the integrated CMDB provides a point of query to multiple federated data sources without becoming a single repository.

Thus, the integrated CMDB can provide a seamless, single point of access to service data distributed across multiple systems, including the CMDB. By using federation as a broker for information exchange behind the scenes, the user does not need to know what federated data source holds which information. Queries are made to the integrated CMDB which acts as a broker to parse and direct the query (or queries) to the appropriate federated data sources and/or its own stored core service data.

A query may be composed dynamically (versus static, pre-defined queries) by a person directly using the query tool in an integrated CMDB interface, or it may be a query sent from within an application that is integrated to the integrated CMDB as a client. In either case, the service data metamodel can structure the query options so the user can preview which data elements and relationships they can ask of the integrated CMDB.

A dynamic query can be a single-source query where the desired information is found in a single federated data source, or it can be a multi-source or distributed query, where different portions of the answer are owned by different federated data sources. When a query seeks information distributed across multiple federated data sources, the query is parsed and submitted automatically to the appropriate federated data sources. The query engine then merges the answers from all the federated data sources into a single query result.

For example, after completion of a planned SAP patch release to an application server, a change manager needs to perform the verification step of the change process. From the change record in a change management application, the change manager issues a single query for the operational performance of the SAP application server and any closely related CIs:

- The query is received by the integrated CMDB which refers to its stored SAP application server CI and identifies several closely related CIs including a database server, load balancer and Lightweight Directory Access Protocol (LDAP) server.
- The query engine looks into the data source registry to see which federated data sources are authoritative for operational performance status for each of these CIs and identifies systems for event management, end user performance, transaction performance and service level agreement (SLA) status.
- The query engine directs performance queries to each of these tools for any out-of-band status for the CIs identified by the integrated CMDB and gathers the results.
- When all the results are gathered by the query engine, a single merged query result is sent back to the change management application with the operational status for all the CIs closely related to the SAP application server. Results are appended to the change record's performance verification field.
- If all the CIs are operating within normal parameters, the change request could be closed. If an anomaly is detected, the change manager now has detailed information encapsulated in the change ticket to provide to the appropriate team for investigation and faster resolution.

Requirement #5: transformation

In our travel example we saw how one common term, "high-speed Internet service," needed to be translated to various representations of Internet service in different hotel systems. Likewise, actionable federation provides data translation between the service data metamodel, client applications and federated data sources, allowing the client to create queries and see results in a common format. The federation service that translates data between clients and federated data sources is referred to as transformation.

For example, a technician in operations using a server configuration tool needs to quickly see all pending requests for change (RFCs) for a given server prior to performing routine maintenance. Yet changes may be stored in two different systems, one depicting RFC status in a single field with "received," "under review," "approved," "in progress" and "complete" as possible values while the other system uses two status fields, one for "open" or "closed" and another field for greater detail. Transformation allows the technician to simply query for "pending" RFCs (as described by the service data metamodel) and get back a single merged set of results from both systems within his server configuration tool. Behind the scenes, the integrated CMDB's federation services are transforming (translating) between the service data metamodel's value of "pending" and the two federated data sources' different representations of "open," "approved" and "in progress."

Transformation can be performed in a partnership between the integrated CMDB (which holds the service data metamodel), federation adapters (where physical data transformation between federated data sources and the service data metamodel is performed) and integrations between client applications and integrated CMDB (where physical data transformation between the applications native data model and the service data metamodel is performed).

The service data metamodel supplies a common structure that data is transformed to and from.

The net result of using the service data metamodel as a common representation for data is normalization. Acting as an Esperanto for applications and repositories across the CMS federation, the service data metamodel provides a common data structure to transform to and from, creating an abstraction layer between the data structures of multiple client applications and federated data sources. With normalization to a common data model, any client can understand any federated data source for any data that has been mapped to the service data metamodel.

Actionable federation advantages

In a CMS context, actionable federation provides several advantages over typical integration strategies such as point-to-point, hardwired, launch in context, and data warehouse and replication strategies:

- **Service context:** By leveraging the service infrastructure context created by discovery and the integrated CMDB, queries can automatically branch out to include related CIs and services. By leveraging the service data metamodel, extended service data in federated data sources are placed in the appropriate service context. The net result is a more intelligent, holistic view of IT information from a business service perspective.
- **Data accuracy:** Because data is accessed on demand directly from the appropriate source of truth (federated data source), even highly dynamic data can be as fresh, accurate and trusted as the federated data source itself.
- **Broader, deeper information:** Because a federated CMS is virtual, even near real-time, highly granular configuration data and management information can be made available in a highly cost-effective manner to any person or tool with CMS access.
- **Wider access:** Because integrations are made to and from the integrated CMDB in a hub-and-spoke manner, it is possible to have many more federated data sources and clients participate in the federation than is feasible with point-to-point connections.
- **Flexibility:** Using the integrated CMDB as a hub between client applications and federated data sources, an IT organization can swap out one federated data source or client application for another and only modify the integration to the integrated CMDB, rather than rebuilding all the hardwired point-to-point integrations that touched the replaced system.

Actionable versus view-only federation

To understand the power of queries, it is helpful to look at actionable federation as a contrast to "view-only federation." View-only federation uses a "launch in context" mechanism to open another application that may have more information about a CI. With launch in context, when a user or application requests federated data about a CI, the federated data source corresponding to that data type is launched in a separate window and potentially opened to a screen containing more information about the CI in the request. With this

approach, no extended service data is reconciled, mapped or exchanged. There is nothing to consume or act on by the client application.

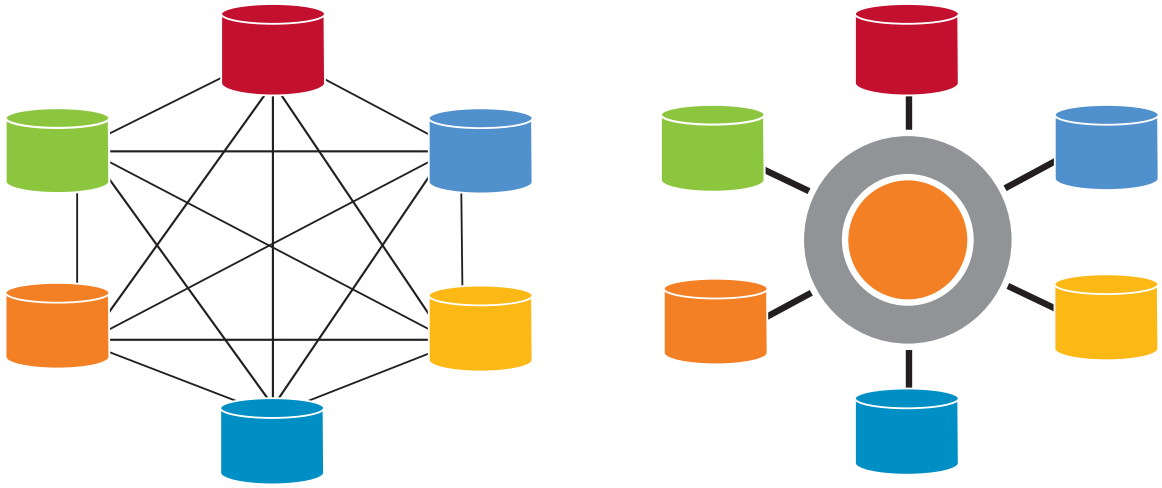
View-only federation can be a useful approach. For example, setup time can be minimal because there is no query syntax or data mapping to configure. Launch in context is also appropriate when the user needs to interact directly with a federated data source's application interface—such as performing diagnostics or logging in to a targeted device.

IT organizations should be aware of the limitations of this approach when the goal is to act on the federated information. The user of the client system is now looking at a second and likely unfamiliar user interface and data model for a separate system, so they must be trained in how to use the federated view to find and understand additional information about the CI. If they wish to transfer information from the federated data source to the client application, they may need to make the appropriate data transformations themselves (in their heads) and then manually cut and paste information. And because users will be logging into sessions and using the federated data source as an application or tool (and not just accessing its data), there may be added complications in managing user licensing for an expanded group of potential federated users.

Given the limitation of launch in context, organizations who wish to make information sharing actionable should consider view-only federation as a supplement to an actionable federation approach.

	View-only federation	**Actionable federation**
Service data metamodel	**No,** only CI data in CMDB is modeled	**Yes,** models CIs in CMDB and data from federated data sources
Data source registry	**Yes**	**Yes**
Dynamic query	**No,** launch in context only	**Yes,** can also include launch in context
CI identity reconciliation	**Yes,** but limited to ability of each federated data source's user interface to present data on requested CIs	**Yes,** queries with parameters understood by federated data source, returns results on as many CIs as requested
Tool transparency	**No,** client application launches view of federated data source in separate window, user must know federated data source interface	**Yes,** query results returned within the user interface and data model of client application
Transformation	**No,** data is view-only in separate window of source application	**Yes,** data transformed via service data metamodel into actionable context for client application

Figure 4. A hub-and-spoke model becomes increasingly more efficient than point-to-point integrations as more systems are added to the CMS federation.



Federation economies of scale

There are times when point-to-point integration makes sense, such as when replicating data from one application to another, or when millisecond response times are needed to support asynchronous transactions between the applications and when a common service context is not needed. However, when three or more systems need to access the same data, federation can offer distinct economies of scale.

Federation adapters to external data sources act as "spokes" off the integrated CMDB "hub," so there is no need to individually define and build each permutation of possible integration combinations. For example, to enable bi-directional data sharing between all combinations of six different tools, traditional point-to-point integration approaches would require 15 integration points. This could potentially include 15 different data transport mechanisms, 15 different CI identity reconciliation setups, 15 different query methods and 15 different translations among data models. In a federated configuration, information sharing for the same 15 permutations is enabled with just six integration points.

Actionable federation as a foundation for automation

Thus far, we have talked about actionable federation primarily in the sense of making information actionable to a user of a client application, e.g., in the user interface and data model context of the decision or action that the user is working on at the time. But delivering information "from anywhere, to anywhere" in the required context also opens the door for more fully automated action, including automatic execution of queries and automatic action taken based on query results.

One of the largest obstacles to interoperability among different management systems is dissimilar data models. Because the service data metamodel and query mechanisms of actionable federation can normalize data to and from a common universal model, queries can be machine-generated, and query results can be machine-consumable. This opens the door to embedding federated queries into automated workflows and for query results to be automatically written into and/or processed by other systems' business logic as part of a larger workflow, either with or without a user's involvement.

Recall that in a previous example a change management user was able to conduct a verification step of an application release change request by querying an application monitoring federated data source for the performance status of the application several hours after the release was executed. Expanding on this scenario:

- Upon completion of a 17-step testing battery, a software quality assurance (QA) application could automatically approve the change request's QA task and leave a record in the change request on what testing steps were completed, avoiding the problem of someone signing off on a change without following the full QA procedure and creating a closed loop audit trail.
- Once the change request is approved, it could trigger the execution of the change to a service automation suite, which could provision a virtual server, install the release, reboot the server and record the change as completed in the change request record.
- During the time the change request specifies that systems are to be taken offline for the application release, it could tell event management tools to suppress events related to the offline systems to avoid unnecessary firefighting and adjust the application's SLA calendar so the offline time won't be counted against it.

Of course, actionable federation alone will not enable all aspects of this level of automation, but the subsystems used for normalized queries provides a foundation of data mapping for such automation to be constructed.

Practical considerations

Implementing a CMS with actionable federation can be far reaching, but an incremental approach can bring value at each step while building a scalable, highly re-usable foundation for cross-domain collaboration. While there isn't a single right answer for all organizations on the right sequence of steps, following are some considerations:

1. **Discover and share application dependency data with a CMDB.** Even before federated data sources are brought online, providing various IT functions with a shared picture of application/infrastructure dependencies will increase the chances that better decisions will be made with regard to the service as a whole. Be sure to choose a combination of application dependency mapping and CMDB that provides the level of granularity, accuracy and currency of data needed to support your key processes and functions.
2. **Identify your authoritative data sources.** Deciding which tools are or should be systems of record for different types of service information may be more of a political issue than a technical one, but the conversation can be valuable, and ultimately necessary. Do you have different teams and tools managing some of the same data? Which have the right data quality and context, as well as ongoing processes and skills for maintaining that data?
3. **Bring a federated data source online.** Identify the most critical near-term pain points and/or initiatives that can benefit from extended service data, and set up a process to use federated queries to one or two authoritative federated data sources for that data. If possible, prioritize where there are two or more tools needing access to the same federated data source to benefit from re-use and economies of scale. Depending on the tools involved, the breadth of data required, and the complexity of the federated data source schema, setting up federated access can take as little as a few hours or days.
4. **Integrate client applications to the integrated CMDB.** Provide a greater level of transparency and actionable use of query results. Enable CMS queries from within the user interface of a client that regularly needs critical federated data source data and filter the options to focus the user on high-value queries.
5. **Enable distributed queries.** Once you determine an existing process or function regularly needs answers to questions that require compilation of data from multiple federated data sources.
6. **Embed queries in automated workflows.** Identify where high-value queries are being regularly repeated with predictable actions taken according to the query results and make them part of a more fully automated action processed either directly by the client system or through an automated runbook tool.

Conclusion

As IT organizations take a closer look at building a sustainable CMS, a federated approach is emerging as the most practical and scalable method. A successful federated approach provides both common service context and coordination of information between multiple data repositories and consumers of this information. Actionable federation can then, in turn, provide the basis for IT organizations to deliver on key initiatives such as IT-business alignment, service management, data center transformation and application quality.

To learn more, visit www.hp.com/software

4AA1-9283ENW, June 2008

Technology for better business outcomes