ServiceNow® Data Separation

A Whitepaper

ServiceNow® Data Separation

A Whitepaper • Ibrahim-Ben Faruhn

Abstract

Domain Separation in ServiceNow often is a complex subject — maybe more complex than it should be. There are different use cases, when you want to use *something like domain separation* to separate data from different parts within your corporation.

The most obvious use case is if your corporation consists of different companies, e.g. a company per country for legal reasons or a company per business area or just legal entities to spread risks.

Another use case could be different departments working within ServiceNow® and of which a few handle sensitive data that shouldn‘t be visible to the other departments.

This Whitepaper explains different approaches and also provides a link to a reference implementation.

About the Author

Ibrahim-Ben Faruhn works as a freelancing ITSM / Business Technical Consultant with 10+ years experience in ITSM and 5+ years experience in ServiceNow®.

While in projects he dug deep into ServiceNow® to always deliver out-of-the-box conformant, upgradable, maintainable, portable, re-usable and future-safe solutions but always managed to translate the technical tongue to a human understandable language and therefor allow sound decisions by higher management.

He started building his ServiceNow® expertise within the German company solid-serVision GmbH being an Architect, Stream Lead, internal technical adviser as well as a quality adviser.

Solid-serVision was well acknowledged across Germany and was recognized by ServiceNow, Inc. while the Knowledge16 as the Quality Leader by having the highest CSAT rating. Solid-serVision was acquired by Accenture in 2017.

Background of this whitepaper

A customer stood in front of a decision on how to separate data for different groups but also on how to share (read-only as well as read-write) certain data between them.

While the ServiceNow® Domain Separation was one way, it is a controversy solution as it seems to be a very heavy solution which adds a lot of organizational overhead. ServiceNow, Inc. explicitly suggests double-checking if Domain Separation is really needed. There are several ways to separate data. Each of which has pros and cons.

VAT ID: DE 309 715 667
IBAN: DE57 1101 0100 1000 0004 52
BIC: SOBKDEBBXXX (solarisBank)

Ibrahim-Ben Faruhn
Müggelseedamm 157
12587 Berlin

Contents

Use Cases

Let‘s start off with cases in which you might want to use a certain kind of separation. Different legal and company-internal requirements also set boundaries which we have to explore.

Levels of separation

When talking of separation, we need to know how „deep“ the separation has to go. This is a first indicator on which approach is feasible for you.

	Description	**Context**
Level 0	Role-model based	Data is visible to all users having appropriate access rights. People with the same role see the same data.
Level 1	Logical Data Separation only	Data is separated using certain criteria on the users as well as on the data. One role model is applied only, but the view is narrowed. Separation is done by OOtB techniques like simple Filters, Business Rules, etc.
Level 2	Logical Data Separation allowing *deviation* in processes	Data is separated using a flag on the users, data and logical elements (e.g. form logic), clearly associating them each to a domain hierarchy. One role model is applied and data is isolated. Separation is done by the Domain Separation Plugin by ServiceNow, Inc. which is deeply integrated into the base ServiceNow® Platform.
Level 3	Logical Data Separation allowing *different* processes	Data is separated using different instances allowing fully different role models and completely different processes. Data is hard to share between instances because of the different role models and processes possible.
Level 4	Physical Data Separation	Like Level 3, but contractually ensured data is living in different physical devices, depending on a legal requirement or the need of security.

Cases for each level of separation

	Use Case	Approaches
Level 0 (Role-model based)	Different departments or companies work in distinct areas (e.g. Finance, HR, Facilities, IT) so that different ACLs and Roles can easily be established to deny or allow access to the specific parts of the system. Read-only rules under certain conditions might be applied to all, roleless or certain users (e.g. caller, owner or watchlist).	• ACLs • Data Policies • UI Policies • View Rules • Predefined Filters
Level 1 (Data separation only)	Different departments or companies cover the same areas, e.g. different / specialised IT departments within the corporate group. The underlying processes and role models are shared between the departments or companies, but data is visible and *accessible* depending on the company or department. Data is (often) shared between the departments or companies as well, e.g. incidents/tasks/cases need to be visible globally and locally or need to be assigned to another corporate company.	• Before/Query Business Rules • Plugin „Domain Separation“
Level 2 (Logical separation)	Like Level 1; the underlying shared processes between the departments or companies are *slightly* different (e.g. different sub-statuses / stages, different mandatory fields, different selection possibilities in reference fields).	• Plugin „Domain Separation“
Level 3 (Full logical separation)	Like Level 2; the underlaying processes are not shared / are fundamentally *different* (e.g. different state model, different role model, different forms) or even incompatible to each other. Data is not likely to be shared between the departments or companies / sharing is clearly defined.	• Separate instances
Level 3 (Physical separation)	Like Level 3 with added need for physical security.	• Separate physical instances and databases

Those cases are another indicator on which approach is the right one for you. When looking at the cases you should not only look at your current or near-future state. You should consider the far future as well. What are possible developments within your company or corporation? Also what will be your role within the organizational construct?

Is a complete separation really needed?

The first thing I would ask (maybe even a couple of times) is if you really need a certain level of separation. With each level, overhead is added. This overhead is at least added on an organizational level but might as well affect commercial and technical aspects. A physical separation requires your ServiceNow® provider to service non-shared hardware which increases hosting costs.

Are you a Service Provider to the different domains?

While Level 0 is more or less easily worked out within a single company as it already has models e.g. allowing access to different physical areas or distinct departments, it gets more complex when different parts of a company or corporation are working on similar things. Also in Level 0, things are likely to stay the same: „you“ operate the ServiceNow® instance, but all in all daily work is the same after implementation. Just the circle of active users increases slightly. More important: *you* keep control of the technology.

When using separation above Level 0, you are inherently sharing a resource in a way that needs more coordination as the resource itself cannot be parted into distinct sections, or so to say *the usage overlaps*. This can be compared to a shared flat. The different parties might want to act independently although they use parts others use as well.

If a separation on Level 3 or 4 is used, there is the possibility, each domain has it‘s own Service Provider for that instance and therefor is easily able to act independently. The shared resource in this case can be the contract to benefit from synergetic effects.

But there is a high possibility on separations above Level 0 that the current team/department/company in charge for the ServiceNow® Instance becomes a kind of Service Provider within the company or corporation — even within the own IT department. The main reason could be that you are the main user of ServiceNow, you already have the expertise, you are simply assigned to be the „operator“ for ServiceNow, or organizational structures suggest that you should lead the platform.

Also in separations above Level 0 it is highly likely you will went from an (internal) Application Provider to a (kind of external) Platform Provider — as seen from the other departments or companies.

You should be aware of the effects of sharing „your“ system to a broader world within your company or corporation and your role should clearly be defined! Without clear structures „your“ system will get messy like the kitchen or bath in a shared flat without an agreed cleaning rota.

Comparing the approaches

ACLs / Data Policies / UI Policies / View Rules / Predefined Filters

This is the simplest method to allow or deny access to data (read-only or read-write). While using ACLs is the strongest option but needing a predefined role model, Data Policies and UI Policies only scratch on the surface by at most denying write access. UI policies can also hide information on forms but do not prevent visibility as such.

With that said, these options are legal if data can be left readable to people having similar positions. They allow a narrow view to the data of interest, but in most cases don't prevent a broader view.

ACLs *can* lock down data, but: if you have a kind of matrix definition of who sees what (e.g. role vs. department), it easily gets quite complex! One reason for that is, that ACLs on the same level, let's say on record level, are OR'd so that just one met ACL on that level already grants access. This makes ACL conditions in such cases very very complex. And if scripts are involved, performance might become an issue as well and at least should be taken into consideration.

Also everyone once ran into the message "Number of rows removed from this list by Security constraints: x" which can only be prevented by (additionally) using predefined filters, e.g. by configuring the corresponding Application Menu entries.

Before/Query Business Rules

Business Rules that run at query time, called Query Business Rules, actually enhance a systems or user's query transparently by adding additional filters in the background. Using this idea, we can pre-filter the user's view on the data without the notice of the user itself.

Such a Business Rule is not just run when viewing a list. It also takes effect on viewing a specific record.

Before Business Rules are run just before data is ultimately sent to the database. They can also abort data manipulation. This is often used to ensure data quality or integrity.

With those capabilities, we can think of a system that, by whatever condition, only provides a narrow view on records and prevents any manipulation outside of that narrow view. The only thing we have to consider is, on which tables that should take effect. Being very careful, one could think to even implement this on task and cmdb_ci level which would make it available on nearly every process within ServiceNow.

Also just two Business Rule per pre-filtered table are needed, making this is a very lightweight solution.

For a reference Implementation see
https://blog.cbc-faruhn.com/tool-data-separation-in-servicenow/

Plugin “Domain Separation”

ServiceNow, Inc. provides a Plugin to provide different levels of separation. In the simplest form you can use this plugin for pure data separation. But it’s power is beyond data: having this plugin activated also activates a mechanism that allows you to deviate certain processual elements, e.g. Business Rules, Choices (incl. State fields), UI policies and Client Scripts, per defined Domain. The plugin is well documented in the ServiceNow® Docs:
https://docs.servicenow.com/csh?topicname=domain-sep-landing-page.html&version=platform

While the “logic deviation part” (called “Delegated Administration”) is enabled by default, you can disable it. Having it enabled adds that sort of overhead which led to that Whitepaper you are currently reading, especially emphasizing on the question “Are you a Service Provider to the different domains?”.

The overhead is caused by the following system behavior:
- Editing components like Business Rules, UI Policies and Client Script in a Domain inherently adds a deviation/variant/version of that component to the Domain, overriding the actual component just for that Domain.
- Admins not paying attention unavoidably create unwanted deviations, making root cause analysis more difficult and adding blind spots to the system.
- There are no Domain-specific admins, which has to lead to close collaboration of the admins if you want to establish admins per Domain.
- Allowing deviation of the processes also means you have to maintain different processes and with each change (globally or in a specific Domain) you have to be aware of the system as a whole keeping “compatibility” in all deviations.
- Although you might think of decentralized work (each Domain works within its own boundaries), the platform is still centralized. All the stakeholders of the different Domains have to pull together.

Separate Instances

Now you might think, giving your Domains an own instance solves a few of the above mentioned obstacles of the Domain Separation plugin. Still there is again this one question: “Are you a Service Provider to the different domains?”

If it’s you that provides the different instances, then it’s you again which has to maintain that additional instance. Also the architectural design quickly gets quite complex:
- Having one Dev & Test, but two different Prod environments makes testing hard and deviations have to be considered at implementation time.
- Having the Dev, Test & Prod chain per Domain allows completely separate development, but also complicates sharing features between the instances as they might develop in complete different directions.

So you see, providing different instances per Domain also requires a strong governance if functionality/technology/innovation is to be shared and you don’t want to support completely different instances.

Comparison Table (Pros and Cons)

	Pros	Cons
ACLs / Data Policies / UI Policies / View Rules / Predefined Filters	• Simple and naive • Overseeable	• Data is mostly just hidden, but still readable • ACLs should be accompanied by predefined filters
Before/Query Business Rules	• Lightweight, yet powerful • Overseeable • Data access is truly prevented	• To be defined per separated (base) table
Plugin "Domain Separation"	• Powerful • Simple configuration • Data access is truly prevented	• Adds organizational overhead • Cannot be uninstalled
Separate Instances	• Allows different processes and role models	• Strong governance required • Shared components have to be kept in sync • Shared Data has to be transferred using integrations

Security considerations

While ACLs are the choice for information security relevant issues, Before/Query Business Rules effectively add the same level of security:

- Business Rules can only be edited (and therefor bypassed) by administrators.
- Before Business Rules can cancel transactions before something is written to the database preventing data manipulation.
- Query Business Rues refine/constrain *every* query to the effective table in a way users cannot bypass, even when retrieving a specific record.

As the Plugin "Domain Separation" really enhances the application layer, this solution seems even securer as ACLs or Before/Query Business Rules, but: it is designed in a way that admins can easily bypass a Domain restriction by a choice integrated in the UI.

In separate instances, an admin from one instance is not necessarily an admin in one of the other instances and therefor cannot bypass security mechanisms in the other Domains. Also users are not shared, which is inherently more secure.