servicenow

ServiceNow Application Development Methodology & Best Practice

Overview 2

Traditional point solutions versus a ServiceNow holistic aPaaS model 2

Team Structure and Onboarding 3

Instance Architecture and Considerations 3

Determine the number of instances needed 3

Application design 4

Development options 4

Global scope 5

Scoped development 5

Introduction to update sets 6

Update set development 6

Update set deployment 6

Introduction to scoped development 7

Scoped application development 7

Scoped application deployment 8

Source code repository, code review, and developer tools 8

High productivity development 8

High control professional development 9

Other development best practices 9

Testing 9

Understand the development use cases 9

Build a new scoped application 9

Build a new shared service/API/library 10

Extend a ServiceNow native application in global scope 11

Extend a third-party scoped application 11

Migration from the global scope into a scoped application 11

ServiceNow recommendations 12

Develop scoped applications 12

Follow delegated development practices 12

Give every process its own scope 13

Avoid changing OOTB files 13

Track development tasks 13

Additional resources 13

Overview

Having a well-defined development methodology is important to any ServiceNow implementation that requires making system changes and moving metadata from one instance to another. It's common to have multiple developers or development streams during an implementation with each developer working on multiple applications or within different scopes. It is also common for a developer to assign the duty of deploying releases to another member of the implementation team. Given the way development teams work, it's important to consider which development path your teams should follow—before they make the first change to your company's system. Choosing the right development practices to follow will help your team ensure their changes work as expected when they deploy them to other instances.

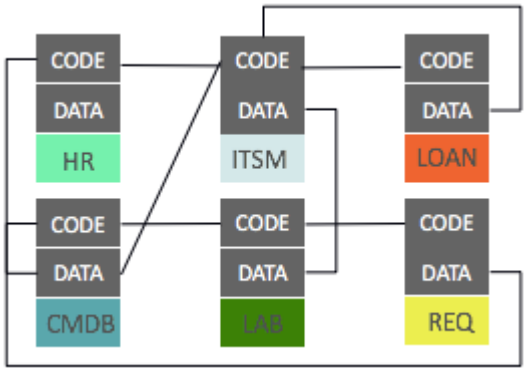
ServiceNow Platform is a powerful tool that enables line-of-business (LOB) process owners, IT Administrators, and professional developers. This is a prescriptive approach on development methodology created by a committee of experts that have worked with various customers and internal groups to give you the best guidance based on experience.

Traditional point solutions versus a ServiceNow holistic aPaaS model

Traditionally, IT has had point solutions for a specific area of use cases, and those were connected via integrations to each other to overcome data dependencies. They were completely segregated—different platforms, databases, and codebases. Integrations are a combinatorically difficult problem. And don't forget all the Microsoft-based apps that aren't integrated with anything, like Outlook, Access, and Excel.

Given that, to find a common dataset or to agree on a shared codebase is one of the most challenging things to do. As a result, your IT team usually ends up trying to reinvent the wheel with each project.

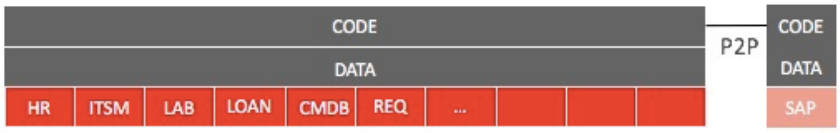
Traditional point solutions



To address those challenges, ServiceNow provides a holistic aPaaS model. All applications span a single platform and have a single codebase, and all applications are natively integrated. Likewise, every application can use external integrations, and your team will only need to govern one global platform.

Some compliance or regulatory requirements may still necessitate separating or protecting data for specific apps. The upcoming chapters will teach you how to draw the line.

Holistic PaaS model



Team Structure and Onboarding

What is the purpose of your team? How big wil it be? Do you want to enable other groups in the enterprise, outside your domain or the domain of IT, and ensure they will not break everyone else? ServiceNow developers usually come in two types:

1. High Productivity – Low code developers, process owners, and administrators
2. High Control – Professional developers

Both types have different desires for working on the ServiceNow Platform. For example, professional developers want to use their own toolsets (IDE, source repository, code review, testing tools, etc.) While most high productivity developers don't want the hassle of configuring a bunch of tools to support development and they require the platform to provide the environment. As a best practice, there should be a strategy on how development teams should operate and this is usually based on who or what is being built on the platform.

It is also recommended that you keep your teams small and as independent as possible. We call this the "Pizza Box" rule, because these types of teams can normally fit around a table, having pizza, and work closely together.

It is also recommended that you delegate development of members of this team to a particular group, that has delegated rights to an application scope (instead of giving admin access to all developers for the entire shared development instance). The requirement of creating a development scope and assigning the correct group is usually a job for the instance (or Platform) administrator and normally an IT responsibility. In some cases this type of delegated assignment can be fully automated and managed by catalog request and workflow fulfillment.

There may be times where you will want to segregate out development further onto separate development instances. This usually occurs when developers are working on the same artifacts at the same time or if developers need to work on different versions of an application at the same time. In this case it is recommended you get more than one development ServiceNow instance. As part of this strategy you will need to reply on source control to manage conflicts and changes. Using Git is our recommended tool for managing this, but the approach to how you use Git is different based on the type of developers you are (mentioned above high productivity or high control). More on this later.

Instance Architecture and Considerations

Determine the number of instances needed

A ServiceNow instance is required to do development and deployment of ServiceNow applications.

It is required that each organization have at least two instances to do development (a subproduction development instance and a production instance). It is recommended as a best practice to use at least three instances whenever possible (a subproduction development instance, a subproduction testing/QA instance, and a production instance). Having at least three instances enables an organization to create a lifecycle of activities from development to production, while also following a testing, validation, and training process.

The types of instances are and should be used as follows:

- Production – This is a customer facing instance critical to application service and delivery. Test all new changes prior to changing this instance. Development is never done here.
- QA/Test – This is a subproduction instance used for regression testing and sometimes training. This instance is cloned over frequently (from the production environment). Development is never done here.
- Development / Sandbox – All development is done on these instances and are sometimes cloned over (from the production environment).
- Personal developer – This instance can be requested on our developer.servicenow.com portal and is used to try new features, training new employees, experimental development, etc. This is instance is easily recoverable.

Application design

The following considerations are recommended for review before developing your application:

- Start with the problem and process you are trying to solve and the personas involved in this application.
- Determine the UI requirements – this will often times dictate which types of developers you will need and how complex your application will be. If the Out-Of-The-Box (OOTB) forms, UI, and list are all that is needed, then we can rely more heavily on the platform features and high productivity development.
- Is the application a fit for the ServiceNow Platform?
- Make sure you are dealing with a new application and not a simple Service Catalog Item Request. Both catalog items and applications (as Record Producers) can be requested (offered) through the Service Catalog, but they are treated much different in terms of complexity.
- Should you extend OOTB tables (such as Task) – as a best practice do not extend a table solely based on the data model. You do not want to extend Task based on which fields the table has, but based on the functionality you need. For example – if your application has a requirement to use the Visual Task Board (VTB) feature of the ServiceNow Platform, then you will need to extend the Task table.
- Modularization is a good concept with an exception – minimize your application dependencies on other app scopes. What this means is that reusable and refactored code and data is recommended within your particular application, but keep external dependencies to a minimum.
- Come up with a naming convention that reflects your company and application function.
- Less is more. Minimize the amount of application files to what is only necessary.

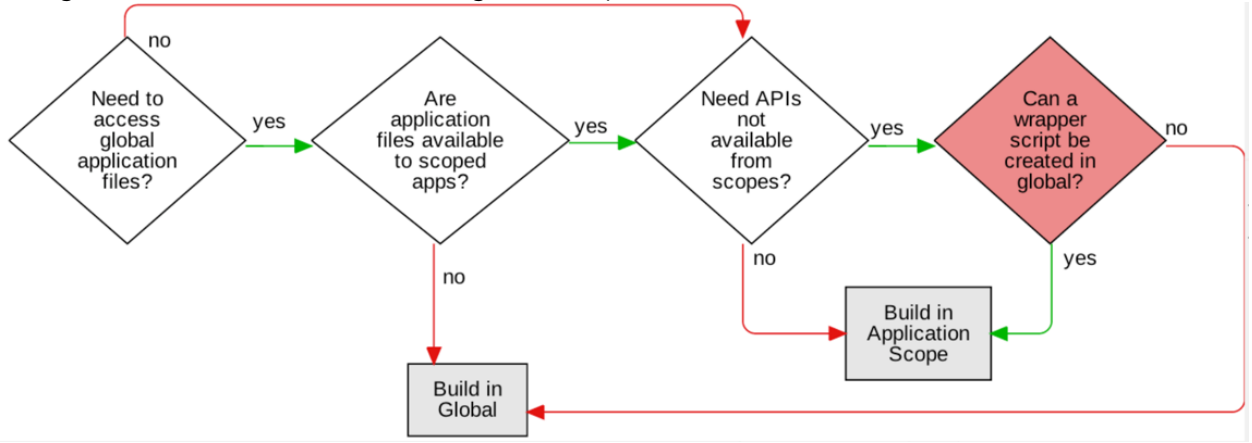
Development options

Development teams can create and scope applications on the ServiceNow Platform in two ways: with a global scope or using scoped development.

Global scope

All development artifacts introduced into this scope can affect the entire system and should be treated with caution. As a best practice working in this scope should be an exception to the rule and not the rule. Heavy use of this development strategy will be more difficult to support and upgrade later on in future releases of ServiceNow.

The general rule of when to build in global scope and how is as follows:



Scoped development

Rather than developing in the global scope, developers should consider scoped development, also called scoped configuration, for any kind of additive modification on the system. This development strategy guarantees design and runtime separation of applications. When communication between applications occurs, developers can define the external APIs, events, and namespaces as well as track the dependencies for all communications.

In this graphic, the applications not built in a global scope are protected and can not interfere with the other applications' code without defined permission. Even an application's administration and development can be delegated to different people in the company without breaking one another. As stated previously, it is recommended as a best practice to delegate administration for a particular scope to a group of developers instead of admin rights to the entire instance given to all developers.

Introduction to update sets

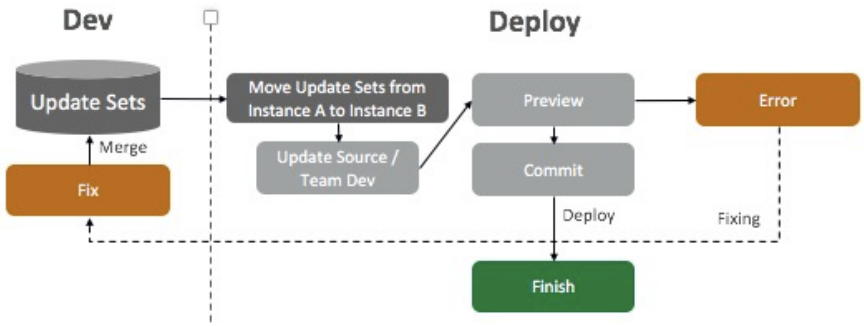
An update set is a group of customizations that developers can move from one instance to another. These customizations are tracked as XML. For example – your team might group a set of changes in an update set called App1. While App1 is marked as the current update set, all changes to it are trakced. Even changes that don't belong to App1 are tracked. It is the developer's responsibility to track these changes according to the development story or context.

Update set development

During the development phase, update sets can be structured in releases, in sprints, per story, or using whatever method your release process requires. Developers are required to use update sets when they develop in the global scope, so they must follow strict development and governance processes to avoid common pitfalls. Teams can batch deploy multiple update sets that are part of the same release. Teams can also report on updates, merge update sets, and compare update sets to ensure the desired changes are ready to move.

Update set deployment

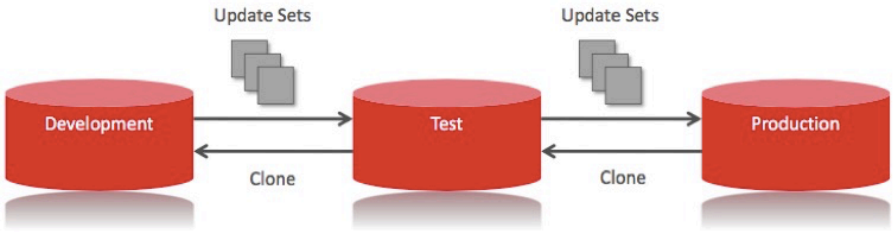
The deployment of update sets involves a few planning steps that developers must fulfill manually on the instance. These are illustrated in the graphic below:



It all starts on a development instance. The developers must close the update sets they plan to move, and they must communicate the names of the update sets they're moving. To avoid common pitfalls, an architect should review the sets. Finally, the team merges them together in specific order according to deployment, avoiding merging the same version of an artifact twice. Alternatively, the team uses the batched update sets process to allow system deployment of multiple update sets as part of one release.

Next, the development team retrieves the merged update set on the target instance and previews it. If the preview shows errors or warnings, the team must fix them by repeating the steps above until no errors are shown or until all are documented. The team then commits the update set to apply all changes on the target instance.

Update set deployment



As the graphic shows, this deployment process must be performed on both test and production.

Introduction to scoped development

The result of using the scoped development approach can be a shippable application, an extension to an application, or even a library or shared service. These applications are collections of files and data that deliver a service and manage business processes. Administrators can also develop and manage custom applications in scopes to meet business needs. When building applications on the ServiceNow platform, application developers can take advantage of existing platform features, such as security access controls, workflows, reporting, and notifications—as well as existing public data, like user and task records.

Scoped application development

Building applications within scopes is the most recommended way to create new content on the ServiceNow Platform. When developers build applications in scopes, they can use this feature-rich set of tools to help them structure, control, and protect their content during the development process:

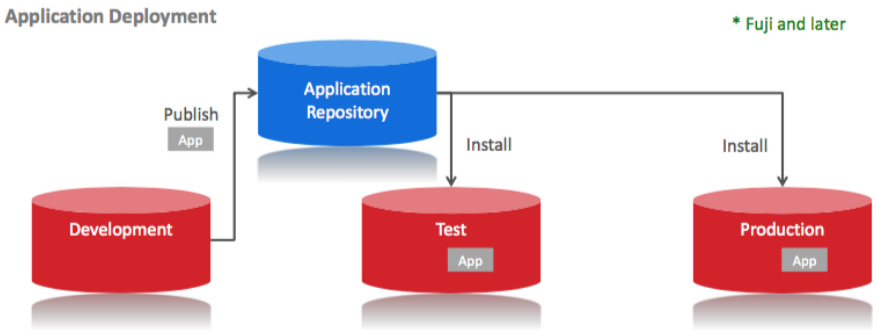
- Development Studio – Provides an IDE-like interface for application developers to work on custom applications in one centralized location; offers a simple way to identify and interact with application files, create files as you develop, and modify existing application files in a tabbed environment; and groups all files belonging to an app together and allows easy access, refactoring, and searching through all files. This IDE is browser based and doesn't require a separate toolset or configuration to setup (beyond the proper ServiceNow access to this workspace).
- Source control – Allows developers to control the changes made to an application over time; provides granular access to your application at certain points in its development, allowing for more detailed editing; and allows companies to work in a collaborative environment and distribute work
- Delegated development – Helps IT and ServiceNow Administrators extend application development to other employees and LOB while maintaining control and governance over the platform, developer privileges, application resources, and data access
- Application repository – Allows development teams to deploy completed applications to other instances as a whole, i.e., they're always deployed 100%; eliminates incremental push; is based on application version; and ServiceNow customers have their own namespaces in the repository that are invisible to everyone else and can only be used within that specific company
- Store – Offers a variety of certified third-party applications and integrations, such as Palo Alto Networks, Boomgar, a QR code generator, and much more, including:
    - Application scoping and access – Application protection through identifying and restricting access to application files and data; assured by the contextual development process, scoped access restrictions, application access settings, and application file protection (to protect the IP)

- ECMA5: The latest and finest JavaScript standard that supports all syntax and behavior features; for all scripts prior to the Helsinki release and not scoped, the ECMA3 compatibility mode will be used

Let's look back to the example about developing some requirements for App1, but instead of developing with a global scope, we'll look at how to create a scoped application. Doing this provides developers with more capabilities and more opportunities to structure, protect, and control the development work.

Scoped application deployment

Deploying applications in scopes is a very sophisticated procedure and includes deploying all application artifacts, like forms, scripts, etc., that are in that scope. That means even a small fix in a script leads to a deployment of all other artifacts, making application deployment management much easier, as it represents the current state of an entire application and is version based. In some cases on premise customers will not have access to an application repository without extra setup, but can still develop in a scoped fashion and deploy a version of entire application as an update set.



Even in larger teams, this process is fast, compact, reliable, and allows faster deployments without any impact to other development streams.

The next graphic illustrates how changes to the ServiceNow Platform are managed. In general, we separate between update sets and scoped applications. Both can be used as separate entities. That means an application can be used without update sets and update sets can be used without global scope applications.

Source code repository, code review, and developer tools

High productivity development

For this type of development it is normally sufficient to use the tools internal to the ServiceNow Platform. These tools include Studio, the ServiceNow Git integration based on application scope on development instances, the ServiceNow application repo for moving applications to QA and Production. This type of development is more restrictive and relies on strong communication, governance, and documentation when planning a release.

High control professional development

For many of our customers and partners, development enviornments outside of the ServiceNow Platform are more flexible and familiar to the current toolset developers deal with day to day. This type of setup has been used successfully and is how many teams within ServiceNow build applications internally. There is no restriction on how ServiceNow interacts with your source code repository because the entire environment (IDE, repo, code review, and some testing) is done outside of the ServiceNow Platform. As most of our code is stored within fields on a table the only integration needed is for outside tools to POST new JavaScript, HTML, or other code into ServiceNow. There are some tools, like UXStorm's File Sync, that help facilitate this type of integration, but ultimately the choice is up to the development team as to how they want to implement these REST API calls to file sync code changes made outside of ServiceNow.

Other development best practices

- Used guided setup
- Create and use OOB extension points
- Include links to documentation as part of your application
- Include usage tracking and diagnostic dashboards as part of your application
- Deploy with logging disabled

Testing

As a best practice use ServiceNow's Automated Testing Framework (ATF) to create a suite of test within your application scope. Test can and should be run on development and QA/Test instances, both before and after deployment of your application, and as part of your upgrade readiness plan when upgrading to a new version of ServiceNow.

Developers should write their own test and will need a specific role to use the ATF. This role should be added to a group and granted through group membership.

Understand the development use cases

Depending on your company's requirements, you might end up extending something existing, creating new content, or publishing an API, which allows certain actions on the managed data.

Each of these scenarios usually falls under one of the following use cases.

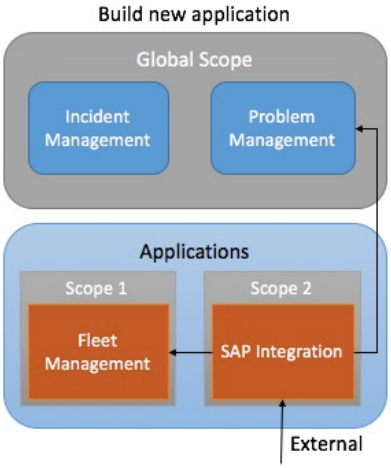
Build a new scoped application

A new application can serve a variety of different use cases, including:

- A new business process not offered on the ServiceNow Platform OOTB.
- A new integration to a business or enterprise application, like a SAP integration.

The graphic below shows an example of a ServiceNow partner company that built two scoped applications—one called Fleet Management and a SAP integration application that supports it—were created to manage a company's car fleet and integrate that process into their major SAP system.

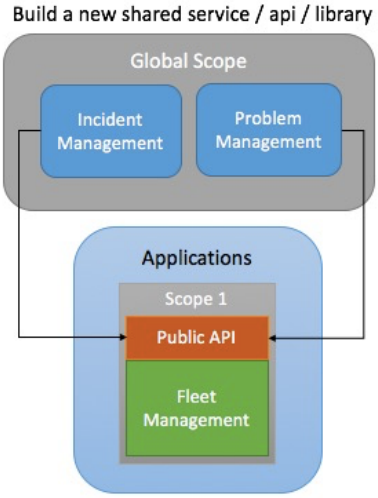
In this example, because the company is a ServiceNow partner, its development team can upload both applications and certify them within the ServiceNow Store to open up another value stream.



Build a new shared service/API/library

After a couple of sprints, a newly made application is ready. At that time, requirements regarding reporting, advanced data analytics, data modification from outside the scope, or cross-process integrations come more into the developers' focus. A good architecture for this sort of data exchange is to offer a public API (like a script include) that can be called from any other scope and can execute several predefined functions and retrieve or modify your data. The graphic below shows an example of what this might look like. The API or library should be part of the existing application scope and can be queried from any other scope. From a developer point of view, it makes it easier to upgrade because only the API needs to be supported, not the whole application process or architecture. This configuration is, of course, the developer's responsibility.

When building a new shared service, API, or library, the incident management service can exchange, modify, or retrieve data from the Fleet Management application without directly accessing it using a common API that only allows specific actions and events to take place on the data. Direct table access might be forbidden, for example, because car or leasing prices are part of the scoped application.



Extend a ServiceNow native application in global scope

A newly created scoped application can be one of the following:

- An OOTB application in the global scope where the data model is extensible
- An OOTB extension point or application file that can be overriden
- Any ServiceNow platform functionality that can be extended, like interfaces, internationalization, UI pages/macros

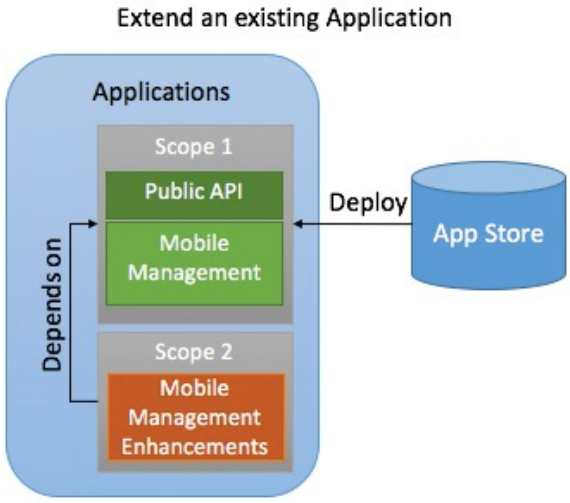
With this approach we want to avoid changing OOTB application files whenever possible. The flexibility is still there to do so for instance administrators, but it creates a certain amount of technical debt that has to be supported.

Extend a third-party scoped application

ServiceNow is a huge ecosystem with a fast growing app store with certified partners creating content. Even some of the newer ServiceNow native applications are published as scoped applications.

In order to use these scoped applications, however, most organizations must do some fine tuning to make the process fit their needs. Depending on the application access settings, you may not be able to modify a specific object. For example, if the tables in a scoped application do not allow configuration, a developer can't add an additional field, business rule, etc. Most process applications do allow configuration because developers want to allow customer specific adjustments. However, with locked-down scoped applications, you can always contact the developer to ask for a public API.

Using the public API, developers can create a new scoped application and add new fields, business logic, etc. Following this process ensures the safety of the original application upgrade. After the upgrade, your developers only need to retest the adjustments they made to the application.



Migration from the global scope into a scoped application

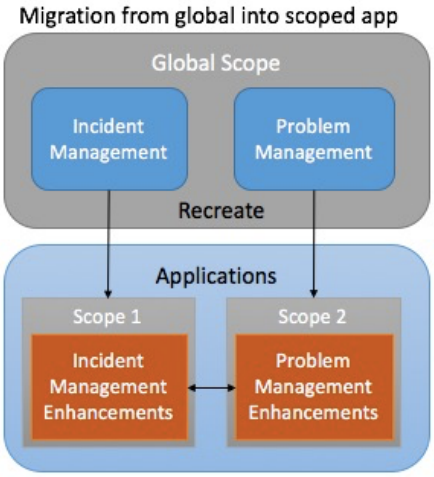
ServiceNow supports working with update sets, and if they're controlled properly, creating an application in the global scope is still a valid option. But it's important to remember that, over time, this method has scalability and maintainability limits. Because of this, you may need to shift your ServiceNow development strategy over time.

Migrating your company's global scope applications might involve one of these two use cases:

- Any (old) update set containing modifications in the global scope
- A ServiceNow implementation prior to the Fuji release

Plan your migration carefully. Developers can simply copy most of the platform objects over to the new scoped application and then deactivate or revert back to OOTB standard of the old (global scope) objects and, later, possibly delete them. There are some objects, like tables and fields, that can't easily be copied. In those cases, test very carefully to discover what it would require to recreate a field and migrate the data from the old field—or to recreate and migrate the whole table. If all of the business logic is already in the scoped application, developers can use the search and replace functionality in Developer Studio to refactor the code connected to the renamed tables or fields.

Over time, all configurations will be in the application and developers can enjoy all of an application's benefits—as well as decreased deployment time.



ServiceNow recommendations

Develop scoped applications

Make working in a scope the rule and changes in global the exception. Working with scoped applications adds more control, enhanced security, and a lean deployment process. It is easier to work in a scope early in your development cycle than going back and changing this later.

Follow delegated development practices

The ServiceNow platform offers a large variety of processes across IT and all other departments. A traditional developer or admin was always able to see and modify everything on the platform, but today's best practice is to delegate development, restricting and controlling administrative access to specific parts of the application. Delegated development increases productivity and accountability and reduces the defects made by unknown cross-process activities. Delegated development also allows multiple development teams to work on the same instance without stepping on each other's toes.

Give every process its own scope

Rather than having one large application scope across multiple processes or logical applications, design each scoped application to be as modular as possible but remember to minimize dependencies that scope has on other application scopes.

Avoid changing OOTB files

Avoid making changes to OOTB application files when possible. There are excpetions to this rule, but a good effort shold be made to use extension points and overrides first.

Track development tasks

In order to manage your releases, sprints, stories, and defects accordingly, ServiceNow recommends the ServiceNow Implementation Methodology (SIM). It combines traditional waterfall and scrum approaches to managing implementation projects. Sim is a set of best practices gathered from hundreds of ServiceNow implementations. The methodology includes standard project and portfolio management functionality , a scrum process pack, and a risk application called RIDAC. With these tools, your team can manage the project from end to end, develop the product with an iterative and agile approach, and control all project risks in one area.

Additional resources

- Scoped applications: https://docs.servicenow.com/bundle/london-application-development/page/build/custom-application/reference/r_CustomApplicationDevelopment.html
- Getting started with update sets: https://docs.servicenow.com/bundle/london-application-development/page/build/system-update-sets/reference/get-started-update-sets.html
- Application and Global scope: https://docs.servicenow.com/bundle/london-application-development/page/build/applications/concept/c_ApplicationScope.html
- Contextual development: https://docs.servicenow.com/bundle/london-application-development/page/build/applications/concept/c_ContextualDevelopmentEnvironment.html
- Protection policies: https://docs.servicenow.com/bundle/london-application-development/page/build/applications/concept/c_ScriptProtectionPolicy.html
- Delegated development: https://docs.servicenow.com/bundle/london-application-development/page/build/applications/concept/c_DelegatedDevelopment.html
- Studio: https://docs.servicenow.com/bundle/london-application-development/page/build/applications/concept/c_ServiceNowStudio.html
- Cross-scope privileges: https://docs.servicenow.com/bundle/london-application-development/page/build/applications/reference/c_CrossScopePrivilegeRecord.html
- Developer portal: http://developer.service-now.com/
- Introduction to scoped applications: https://community.servicenow.com/community/develop/app-publisher/blog/2015/01/26/scoped-applications--introduction
- Background and philosophy of scoped applications: https://community.servicenow.com/community?id=community_blog&sys_id=801e6e2ddbd0dbc01dcaf3231f9619d8&view_source=searchResult
- Scoped applications and client scripts: https://community.servicenow.com/community?id=community_blog&sys_id=788c66e1dbd0dbc01dcaf3231f961969&view_source=searchResult