# ServiceNow: Linking Records to Employee Data

## Introduction to Employee Record Linking in ServiceNow

This project aims to enhance the functionality and reporting capabilities within our ServiceNow instance by establishing a robust link between various records and employee data. In today's data-driven environment, understanding the context behind service requests, incidents, changes, and other operational records is paramount. Without direct ties to the individuals involved, reporting and analysis can be superficial, leading to inefficiencies and missed opportunities for process improvement. By integrating employee attributes directly into relevant ServiceNow records, we can unlock deeper insights, improve data accuracy, and ultimately create a more streamlined and user-friendly experience for both end-users and administrators.

### The Business Need for Employee Data Integration

The core business need driving this initiative stems from the limitations of our current data management practices within ServiceNow. Often, when reviewing reports or investigating specific records, a significant amount of manual effort is required to cross-reference information with our human resources systems or internal knowledge bases. This is time-consuming, prone to errors, and hinders our ability to quickly grasp the full impact of a particular item. For instance:

* **Reporting & Analysis:** Without linking records to specific employees, it is challenging to analyze trends based on department, location, or management structure. This limits our ability to identify systemic issues within specific business units or to assess the impact of services on different employee populations.
* **Data Accuracy & Consistency:** Relying on manual entry or separate data sources for employee information can lead to discrepancies. Linking directly ensures that the most up-to-date employee details are consistently associated with relevant ServiceNow records.
* **User Experience:** For end-users and support staff, having immediate access to contextual employee information within a record (e.g., the affected user's department, manager, or contact details) significantly improves efficiency and reduces the need for follow-up inquiries.
* **Process Efficiency:** Automating the association of employee data reduces the need for manual data entry and cross-referencing, freeing up valuable time for IT staff to focus on more strategic tasks.

### Leveraging the User (sys\_user) Table

ServiceNow maintains a central repository for all user information in its **User (sys\_user)** table. This table is rich with attributes such as employee ID, name, department, location, manager, job title, and contact information. Our project will strategically leverage this existing table by creating relationships from other ServiceNow record types (e.g., Incident, Request, Change Request) to the **User (sys\_user)** record of the individual associated with that item. This approach ensures that we are utilizing a single, authoritative source for employee data, maintaining data integrity and simplifying future maintenance.

### Benefits of Centralized Employee Information

By centralizing employee information and linking it directly to operational records, we stand to gain several significant benefits:

* **Enhanced Contextual Understanding:** Quickly understand who is affected by an issue, who is requesting a service, or who is responsible for a change, along with their organizational context.
* **Improved Reporting Accuracy:** Generate more precise and insightful reports based on employee attributes, enabling better decision-making and resource allocation.
* **Streamlined Workflows:** Reduce manual data lookup and entry, accelerating resolution times and improving overall operational efficiency.
* **Greater Data Integrity:** Ensure that employee information associated with ServiceNow records is accurate and up-to-date by directly referencing the primary user source.

This foundational step sets the stage for a more intelligent and efficient ServiceNow environment, paving the way for deeper analysis and more targeted support.

## Understanding the ServiceNow User (sys\_user) Table

At the heart of ServiceNow's user management and employee data integration lies the **User (**sys\_user**)** table. This table serves as the central repository for all individuals who interact with the ServiceNow platform, whether as end-users, fulfillers, administrators, or external collaborators. Understanding its structure and the wealth of information it contains is fundamental to successfully linking records and extracting meaningful insights for reporting and administrative efficiency.

### Purpose and Structure of the sys\_user Table

The sys\_user table is a foundational data structure within ServiceNow. Its primary purpose is to store comprehensive details about each user, enabling ServiceNow to manage access, personalize experiences, and facilitate various operational processes. Each record in this table represents a unique user and contains a predefined set of fields designed to capture essential information.

### Key Fields for Reporting and Analysis

For the purpose of this project, several fields within the sys\_user table are particularly crucial. These fields provide the necessary context to understand the "who" and "where" behind the data we manage in ServiceNow:

* **User ID (**user\_name**):** This is the unique identifier for a user within ServiceNow. It's often the username used for logging into the platform. While essential for system operations, for reporting, we typically use more human-readable fields.
* **Name (**name**):** The full name of the employee. This is vital for identifying individuals directly within reports and for user-friendly display in associated records.
* **Email (**email**):** The primary email address of the user. This is critical for communication, notifications, and often serves as another unique identifier when integrating with other systems. It's invaluable for reporting on communication patterns or user outreach.
* **Department (**department**):** This field links to the **Department (**cmn\_department**)** table and specifies the organizational unit the user belongs to. Linking records to a user's department allows for powerful reporting on how IT services, incidents, or requests impact different business units. This can highlight departmental needs or uncover systemic issues within specific areas of the organization.
* **Cost Center (**cost\_center**):** Similar to Department, this field links to the **Cost Center (**cost\_center**)** table. It provides financial context, indicating which cost center the user is associated with. This is invaluable for chargeback reporting, budget analysis, and understanding the financial implications of IT service consumption by different parts of the business.
* **Location (**location**):** This field links to the **Location (**cmn\_location**)** table and indicates the physical or logical location of the user. Reporting by location can help identify geographically specific issues, manage resources in different offices, or understand service demand patterns across various sites.
* **Manager (**manager**):** This field creates a reference back to another record in the sys\_user table, identifying the user's direct manager. Linking incidents or requests to a user's manager can streamline communication for escalations, facilitate approvals, and provide insights into how management structures influence service adoption or support needs.

### Utilizing sys\_user Fields for Enhanced Reporting

The true power of the sys\_user table emerges when these fields are used in conjunction with other ServiceNow data. For example:

* When an incident is reported, linking it to the affected user allows us to automatically populate fields like 'Department', 'Location', and 'Manager' on the incident form.
* Subsequent reports can then be generated to show "Number of Incidents by Department," "Average Resolution Time by Location," or "High Priority Incidents by Manager."
* This granular level of detail moves reporting from a general overview to actionable intelligence, enabling targeted improvements and more effective resource management.

### The Importance of Data Hygiene

The effectiveness of any reporting or process improvement initiative hinges on the quality of the underlying data. Therefore, maintaining data hygiene within the sys\_user table is paramount. This involves ensuring that:

* User records are accurate and up-to-date.
* Fields like Department, Location, and Manager are correctly populated and regularly reviewed.
* Duplicate user records are minimized or eliminated.
* Data is synchronized with authoritative sources (e.g., HR systems) to maintain consistency.

A clean and well-maintained sys\_user table ensures that the insights derived from linked records are reliable and trustworthy.

## Project Scope and Objectives

This project is focused on establishing a direct and meaningful link between key operational records within ServiceNow and the corresponding employee data stored in the sys\_user table. The primary goal is to enrich these records with contextual employee attributes, thereby enhancing reporting capabilities, improving administrative efficiency, and providing deeper insights into service delivery and user impact.

### Scope of Record Linking

The initial phase of this project will concentrate on integrating employee data into the following critical record types:

* **Incidents:** Records pertaining to unplanned interruptions or reductions in the quality of an IT service. Linking incidents to the affected user will provide immediate context regarding their department, location, and management structure.
* **Requested Items:** Records representing a request made by an end-user for a new service or item, typically fulfilled through the Service Catalog. Linking these items to the requester allows for analysis of service demand by organizational unit or location.
* **Change Requests:** Records detailing planned modifications to IT services or infrastructure. While the primary focus might be on the requester or implementer, linking to the user impacted by the change (if applicable) can provide valuable reporting data.

This phased approach allows for focused development and testing, ensuring a robust and reliable integration before expanding to other record types.

### Employee Details to be Integrated

To achieve the project's objectives, the following specific employee details, sourced directly from the sys\_user table, will be linked and made accessible within the scoped records:

* **Department:** The organizational unit or business unit to which the employee belongs. This will be populated from the department field in the sys\_user table.
* **Location:** The physical or logical location associated with the employee. This will be populated from the location field in the sys\_user table.
* **Manager:** The direct manager of the employee, referencing another record within the sys\_user table. This will be populated from the manager field.
* **Cost Center:** The financial reporting unit the employee is associated with, populated from the cost\_center field.

These attributes are chosen for their direct relevance to common reporting needs, workflow analysis, and administrative oversight.

### Project Objectives

The successful completion of this project will yield the following key objectives:

* **Reduce Manual Data Entry:** Automate the population of employee-related fields on relevant records, minimizing the need for manual input and reducing associated errors.
* **Enhance Report Accuracy and Depth:** Enable the creation of more granular and insightful reports by leveraging employee attributes such as department, location, and manager, leading to better-informed decision-making.
* **Provide a Unified View of Service Delivery:** Offer a holistic understanding of how IT services, incidents, and changes impact different segments of the employee base within the organization.
* **Streamline Workflows:** Facilitate quicker identification of affected parties, managers for escalations, and relevant departments, thereby speeding up resolution and approval processes.
* **Improve Data Integrity:** Ensure consistency and accuracy of employee information associated with ServiceNow records by directly linking to the authoritative sys\_user table.

## Technical Approach: Core Concepts and Design

To effectively link ServiceNow records to employee data and leverage associated attributes for enhanced reporting, we will employ several core ServiceNow functionalities and design principles. This section outlines the fundamental concepts and the technical approach that will be utilized, focusing on how ServiceNow's relational database structure and scripting capabilities enable this integration.

### Leveraging Reference Fields

At the core of relating different data points within ServiceNow is the concept of **Reference Fields**. A reference field stores a unique identifier (the sys\_id) of a record in another table. This creates a direct, relational link between records. For this project, the primary method of linking will be by introducing a reference field on our target tables (e.g., incident, sc\_req\_item) that points to the **User (**sys\_user**)** table. This means that instead of manually entering employee details, we will select an employee from a dynamically populated list, and the system will store a reference to that employee's record.

#### The ref\_auto\_completer Attribute

To optimize the user experience when selecting an employee from the sys\_user table, we will utilize the ref\_auto\_completer attribute. This attribute, when applied to a reference field, enables auto-completion functionality. As a user types characters into the reference field, the system intelligently queries the referenced table (in our case, sys\_user) and presents a list of matching records. This significantly speeds up the selection process and reduces errors compared to a simple dropdown or manual lookup. We will configure this attribute to search on key fields within the sys\_user table, such as 'Name', 'User ID', and 'Email', ensuring users can quickly find the correct employee.

### Data Model Adjustments for Target Tables

To incorporate employee details, we will need to make specific adjustments to the data model of the target ServiceNow tables.

#### Introducing New Fields

For each employee attribute we wish to make readily available for reporting and display, we will create a new field on the respective target table. For instance:

* On the incident table, we will add fields such as 'Affected Department', 'Affected Location', and 'Affected Manager'.
* Each of these new fields will be configured as a **Reference Field**.
* The 'Affected Department' field will reference the sys\_user.department field (which in turn points to the cmn\_department table).
* The 'Affected Location' field will reference the sys\_user.location field (pointing to the cmn\_location table).
* The 'Affected Manager' field will reference the sys\_user.manager field (which is a self-reference within the sys\_user table).

#### Data Population Strategy

The primary reference field, linking the target record to a specific user, will be the 'Affected User' or similarly named field. Once this field is populated, the associated employee details will be automatically referenced. The actual population of the newly created 'Affected Department', 'Affected Location', and 'Affected Manager' fields will be handled by business rules or client scripts, as detailed below.

### Populating Associated Employee Details

Once a record is linked to a specific user via the primary reference field (e.g., 'Affected User' on an Incident), we need a mechanism to automatically populate the other employee detail fields (Department, Location, Manager, etc.) on the form. This is typically achieved using one of two primary ServiceNow scripting methods:

#### 1. Display Business Rules

Display Business Rules run on the server-side when a record is displayed in a form. They are ideal for populating fields on the form with data derived from related records, such as the sys\_user table. We can configure a display business rule on the target tables (e.g., incident) that triggers when the 'Affected User' field is changed or the form loads. This business rule will:

* Check if the 'Affected User' field is not empty.
* If populated, query the sys\_user table using the sys\_id from the 'Affected User' field.
* Retrieve the values for 'Department', 'Location', and 'Manager' from the queried user record.
* Set the values of the corresponding fields on the target record (e.g., 'Affected Department', 'Affected Location', 'Affected Manager') using the g\_scratchpad object. The g\_scratchpad allows server-side scripts to pass data to client-side scripts, making it readily available on the form.

This approach is efficient for populating fields that are displayed directly on the form.

#### 2. Client Scripts (onChange and onLoad)

Alternatively, or in conjunction with display business rules, Client Scripts can be used. An onChange client script can monitor changes to the 'Affected User' reference field. When a user is selected, this script can:

* Retrieve the sys\_id of the selected user.
* Use asynchronous GlideAjax calls to a Script Include (a server-side script) to fetch the user's department, location, and manager details.
* Update the corresponding fields on the form using the g\_form.setValue() and g\_form.addOption() methods.

An onLoad client script can be used in conjunction with a display business rule that populates g\_scratchpad to populate the fields when the form initially loads, ensuring the data is present immediately.

### Choosing the Right Approach

For simplicity and performance, especially when many fields need to be populated from a single referenced record, a **Display Business Rule** combined with g\_scratchpad is often the preferred method for initial population. Client scripts can then be used for dynamic updates or conditional logic. This ensures that the required employee details are consistently and automatically populated on the record forms, providing the necessary context for improved reporting and workflow efficiency.

## Implementation Steps: Configuration and Development

This section provides a detailed, step-by-step guide to implementing the employee record linking solution within ServiceNow. We will cover the process of identifying target tables, creating necessary reference fields, configuring their attributes, and developing scripts to ensure seamless data population and enrichment. The objective is to transform our ServiceNow instance into a more context-aware platform, enabling deeper insights and streamlined operations.

### 1. Identifying Target Tables

The first crucial step is to identify which ServiceNow tables will benefit most from being linked to employee data. Based on our project objectives, we will initially focus on tables that represent common interactions and require user context for reporting and analysis. These typically include:

* **Incident (**incident**):** To understand the impact of incidents on specific users, departments, and locations.
* **Requested Item (**sc\_req\_item**):** To analyze service demand trends across different organizational units and locations.
* **Change Request (**change\_request**):** To understand the impact of changes on users or to track changes initiated by specific departments.
* **Problem (**problem**):** To associate known errors with the users or departments most affected.
* **Service Request (**sc\_request**):** While sc\_req\_item is more granular, linking the overall request can also be beneficial for high-level analysis.

For the purpose of this guide, we will use the incident table as our primary example for demonstrating the configuration and development steps. The same principles can be applied to other identified tables.

### 2. Adding New Reference Fields

For each target table, we need to establish a direct link to the **User (**sys\_user**)** table. This is achieved by adding a new field that acts as a reference. We will also add fields to store specific employee attributes that we want to be readily visible and reportable directly on the target record.

#### Step 2.1: Adding the Primary User Reference Field

On the target table (e.g., incident), a new field will be added to reference the user who is most relevant to the record. This could be the 'Affected User', 'Requested For', 'Opened By', or 'Assigned To', depending on the specific context needed. For this guide, let's assume we are adding an 'Affected User' field to the incident table.

1. Navigate to **System Definition > Tables**.
2. Find and open the incident table record (com.glide\_form).
3. Scroll down to the "Columns" related list.
4. Click the **New** button to create a new column.
5. Configure the new column as follows:
   * **Table:** Incident [incident]
   * **Type:** Reference
   * **Column label:** Affected User
   * **Name:** affected\_user (ServiceNow will automatically add 'u\_' prefix if it's a custom field, e.g., u\_affected\_user)
   * **Reference:** User [sys\_user]
   * **Reference qualifier:** Simple (or None, depending on needs)
6. Click **Submit**.

#### Step 2.2: Adding Fields for Employee Attributes

To easily report on department, location, and manager without having to perform a lookup every time, we will add fields on the incident table that will be populated by the 'Affected User's' data. These fields will also be reference fields, but they will reference the specific attribute's table (Department, Location, etc.).

1. On the incident table's "Columns" related list, click **New** again.
2. Configure the 'Department' field:
   * **Table:** Incident [incident]
   * **Type:** Reference
   * **Column label:** Affected Department
   * **Name:** u\_affected\_department
   * **Reference:** Department [cmn\_department]
3. Click **Submit**.
4. Repeat the process to add:
   * **Column label:** Affected Location
   * **Name:** u\_affected\_location
   * **Reference:** Location [cmn\_location]
5. Click **Submit**.
6. Repeat the process to add:
   * **Column label:** Affected Manager
   * **Name:** u\_affected\_manager
   * **Reference:** User [sys\_user]
7. Click **Submit**.

**Note:** While we are adding reference fields for Department and Location, the primary goal is to \*display\* the names of these entities. We will use scripting to populate these fields with the correct values based on the selected 'Affected User'. The 'Affected Manager' field is correctly a reference to the sys\_user table itself, allowing for hierarchical reporting.

### 3. Configuring Reference Field Attributes

Once the fields are created, we need to configure their attributes to optimize usability and ensure data integrity.

#### Step 3.1: Adjusting Field Attributes for 'Affected User'

1. Navigate back to the incident table's "Columns" related list and open the newly created u\_affected\_user field.
2. In the "Attributes" field, add ref\_auto\_completer=true. This enables the auto-completion feature, allowing users to quickly find users by typing their name or username. You might also want to add ref\_auto\_completer\_input=true for a more robust auto-completion experience.
3. Optionally, set the **Mandatory** checkbox to **true** if an 'Affected User' must be selected for every incident.
4. Add a **Tooltip** for clarity, e.g., "Select the user directly affected by this incident."
5. Click **Update**.

#### Step 3.2: Configuring Attributes for Employee Attribute Fields

For the 'Affected Department', 'Affected Location', and 'Affected Manager' fields, we want to ensure they are read-only on the form itself, as their values will be populated by scripts.

1. Open the u\_affected\_department field record.
2. Go to the "Dictionary" entry for this field (usually accessed by right-clicking the column header or via the "Attributes" tab).
3. In the "Attributes" field, add readonly.
4. Click **Update**.
5. Repeat this process for u\_affected\_location and u\_affected\_manager, adding the readonly attribute.

**Note on UI Policies/Client Scripts for Read-Only:** While adding the 'readonly' attribute in the dictionary works, a more flexible approach for managing field visibility and states is often using UI Policies or Client Scripts. For example, a UI Policy could set these fields to read-only based on the form view, or a Client Script could handle the read-only state dynamically.

### 4. Creating Scripts for Data Population

With the fields in place, we need to implement logic to automatically populate the 'Affected Department', 'Affected Location', and 'Affected Manager' fields whenever the 'Affected User' field is changed.

#### Step 4.1: Using a Display Business Rule (Recommended for initial population)

A Display Business Rule is ideal for populating fields on a form when the record loads or when a referenced field changes. It runs on the server and makes data available to the client via g\_scratchpad.

1. Navigate to **System Definition > Business Rules**.
2. Click **New**.
3. Configure the Business Rule:
   * **Name:** Populate Affected User Details (Incident)
   * **Table:** Incident [incident]
   * **Advanced:** Checked
   * **When to run:**
     + **When:** display
     + **Order:** 100 (or adjust as needed)
     + **Filter Conditions:** Affected User is not empty
4. In the "Advanced" tab, use the following script:

(function executeRule(current, previous /\* undefined when script is responsible for new record creation \*/) {  
  
 // Get the sys\_id of the affected user  
 var affectedUserSysId = current.getValue('u\_affected\_user');  
  
 // If an affected user is selected, fetch their details  
 if (affectedUserSysId) {  
 var userGr = new GlideRecord('sys\_user');  
 if (userGr.get(affectedUserSysId)) {  
 // Set values to g\_scratchpad for client-side access  
 g\_scratchpad.affectedDepartment = userGr.getValue('department');  
 g\_scratchpad.affectedLocation = userGr.getValue('location');  
 g\_scratchpad.affectedManager = userGr.getValue('manager');  
 g\_scratchpad.affectedUserDisplayName = userGr.getDisplayValue('name'); // Optional: for display purposes  
 } else {  
 // Clear scratchpad variables if user not found  
 g\_scratchpad.affectedDepartment = '';  
 g\_scratchpad.affectedLocation = '';  
 g\_scratchpad.affectedManager = '';  
 g\_scratchpad.affectedUserDisplayName = '';  
 }  
 } else {  
 // Clear scratchpad variables if affected user is cleared  
 g\_scratchpad.affectedDepartment = '';  
 g\_scratchpad.affectedLocation = '';  
 g\_scratchpad.affectedManager = '';  
 g\_scratchpad.affectedUserDisplayName = '';  
 }  
  
 })(current, previous);

Click **Submit**.

#### Step 4.2: Using an onLoad Client Script

Now, we need a Client Script that runs when the form loads (or when the Business Rule populates g\_scratchpad) to set the values of our newly created fields.

1. Navigate to **System Definition > Client Scripts**.
2. Click **New**.
3. Configure the Client Script:
   * **Name:** Set Affected User Details (Incident)
   * **Table:** Incident [incident]
   * **UI Type:** desktop
   * **Type:** onLoad
4. Use the following script:

function onLoad() {  
 // Check if g\_scratchpad variables are populated by the display Business Rule  
 if (g\_scratchpad.affectedDepartment) {  
 g\_form.setValue('u\_affected\_department', g\_scratchpad.affectedDepartment);  
 } else {  
 g\_form.setValue('u\_affected\_department', ''); // Clear if no user selected or found  
 }  
  
 if (g\_scratchpad.affectedLocation) {  
 g\_form.setValue('u\_affected\_location', g\_scratchpad.affectedLocation);  
 } else {  
 g\_form.setValue('u\_affected\_location', ''); // Clear if no user selected or found  
 }  
  
 if (g\_scratchpad.affectedManager) {  
 g\_form.setValue('u\_affected\_manager', g\_scratchpad.affectedManager);  
 } else {  
 g\_form.setValue('u\_affected\_manager', ''); // Clear if no user selected or found  
 }  
  
 // Optionally, if you want to ensure the primary 'Affected User' field is populated if the Business Rule cleared it  
 // This is less common for onLoad, more for onChange, but illustrates the concept.  
 // If g\_form.getValue('u\_affected\_user') is empty and g\_scratchpad.affectedUserDisplayName is available,  
 // you might want to retrieve the sys\_id, which requires an async GlideAjax call or different logic.  
 // For simplicity, we rely on the display BR and the user selecting the 'Affected User'.  
 }

Click **Submit**.

#### Step 4.3: Using an onChange Client Script (Optional but Recommended for dynamic updates)

While the onLoad script works when the form loads, an onChange script ensures that if the 'Affected User' is changed \*after\* the form has loaded, the associated details are updated immediately without a page refresh.

1. Navigate to **System Definition > Client Scripts**.
2. Click **New**.
3. Configure the Client Script:
   * **Name:** Update Affected User Details on Change (Incident)
   * **Table:** Incident [incident]
   * **UI Type:** desktop
   * **Type:** onChange
   * **Field name:** Affected User [u\_affected\_user]
4. Use the following script:

function onChange(control, oldValue, newValue, isLoading, isTemplate) {  
 if (isLoading || newValue === '') {  
 // Clear fields if the Affected User is cleared  
 g\_form.setValue('u\_affected\_department', '');  
 g\_form.setValue('u\_affected\_location', '');  
 g\_form.setValue('u\_affected\_manager', '');  
 return;  
 }  
  
 // The Display Business Rule already runs on change if the field is modified.  
 // However, to ensure immediate visual feedback \*without\* a page refresh,  
 // we can re-trigger the logic or use GlideAjax for more complex scenarios.  
 // For this setup, the onLoad script (triggered by the BR's g\_scratchpad update)  
 // is often sufficient if the BR fires on the change event.  
 // If not, or for guaranteed immediate update, we can call a Script Include via GlideAjax.  
  
 // --- Using GlideAjax for real-time update (more robust) ---  
 var ga = new GlideAjax('UserDetailFetcher'); // Name of the Script Include  
 ga.addParam('sysparm\_name', 'getUserDetails'); // Function in Script Include  
 ga.addParam('sysparm\_user\_id', newValue); // Pass the selected user's sys\_id  
 ga.getXML(setUserDetailFields);  
  
 function setUserDetailFields(response) {  
 var answer = response.responseXML.documentElement.getAttribute("answer");  
 var userDetails = JSON.parse(answer);  
  
 if (userDetails) {  
 g\_form.setValue('u\_affected\_department', userDetails.department);  
 g\_form.setValue('u\_affected\_location', userDetails.location);  
 g\_form.setValue('u\_affected\_manager', userDetails.manager);  
 } else {  
 // Clear fields if details couldn't be fetched  
 g\_form.setValue('u\_affected\_department', '');  
 g\_form.setValue('u\_affected\_location', '');  
 g\_form.setValue('u\_affected\_manager', '');  
 }  
 }  
 // --- End GlideAjax ---  
  
 // NOTE: If using Display BR + onLoad script, the Business Rule should be configured to run on 'update' and potentially 'insert'  
 // and the onLoad script will pick up the changes. The onChange script provides more immediate feedback.  
 // For simplicity in initial setup, relying on Display BR + onLoad is often enough.  
 // The example above demonstrates the GlideAjax approach for immediate updates.  
 }

Click **Submit**.

#### Step 4.4: Creating the Script Include for GlideAjax (if used in onChange)

If you choose to use the GlideAjax approach in the onChange script, you'll need a Script Include.

1. Navigate to **System Definition > Script Includes**.
2. Click **New**.
3. Configure the Script Include:
   * **Name:** UserDetailFetcher
   * **API Name:** global.UserDetailFetcher
   * **Client callable:** Checked
   * **Accessible from:** All application scopes
4. Use the following script:

var UserDetailFetcher = Class.create();  
 UserDetailFetcher.prototype = Object.extendsObject(AbstractAjaxProcessor, {  
  
 getUserDetails: function() {  
 var userId = this.getParameter('sysparm\_user\_id');  
 var userDetails = {};  
  
 if (userId) {  
 var userGr = new GlideRecord('sys\_user');  
 if (userGr.get(userId)) {  
 userDetails.department = userGr.getValue('department');  
 userDetails.location = userGr.getValue('location');  
 userDetails.manager = userGr.getValue('manager');  
 }  
 }  
 // Return JSON stringified object  
 return JSON.stringify(userDetails);  
 },  
  
 type: 'UserDetailFetcher'  
 });

Click **Submit**.

### 5. Testing and Verification

After implementing these steps, thorough testing is essential:

* Create a new Incident record.
* Select an 'Affected User' using the auto-completion.
* Verify that the 'Affected Department', 'Affected Location', and 'Affected Manager' fields are automatically populated with the correct information.
* Change the 'Affected User' to another user and confirm that the fields update accordingly (either via the onLoad script picking up the Business Rule change, or immediately via the onChange script).
* Clear the 'Affected User' field and verify that the associated detail fields are also cleared.
* Test the functionality across different browsers and user roles.

These steps provide a solid foundation for linking employee data to your ServiceNow records, enabling richer reporting and more efficient workflows.

## Reporting and Analytics Enhancements

The integration of employee data directly into ServiceNow records transforms our ability to generate meaningful reports and gain actionable insights. By linking incidents, requests, and other operational items to the sys\_user table, we unlock a wealth of contextual information that was previously difficult or impossible to access efficiently. This section details how these enhancements improve reporting capabilities and demonstrates the types of powerful analyses now available through the Now Platform's robust reporting engine.

### Leveraging Employee Attributes for Deeper Insights

Having employee attributes such as Department, Location, and Manager readily available on service records provides crucial context that significantly enhances the value of our reporting. Instead of just knowing \*that\* an incident occurred, we can now understand \*who\* it affected and their organizational placement. This allows for:

* **Understanding Service Impact:** Analyze how IT services, incidents, or requests are utilized and impact different business units. For example, identifying which departments experience the highest volume of critical incidents can highlight areas needing targeted support or process improvements.
* **Resource Allocation and Planning:** Reports based on location can inform decisions about IT support staffing, infrastructure deployment, or rollout strategies for new services in specific geographical areas.
* **Identifying Trends by Management Structure:** Understanding which managers' teams are experiencing higher support volumes can help identify potential training needs, workflow bottlenecks, or specific project-related issues that require managerial attention.
* **Financial Analysis:** When linked to Cost Centers, service consumption data can be used for chargeback initiatives, budget forecasting, and understanding the financial implications of IT service delivery across the organization.
* **Root Cause Analysis:** By correlating incidents or problems with the departments or locations of affected users, we can more effectively pinpoint systemic issues or environmental factors contributing to recurring problems.

The direct access to these attributes eliminates the need for manual data correlation or complex join operations across disparate systems, making reporting faster, more accurate, and far more insightful.

### Building Reports and Dashboards in ServiceNow

The Now Platform's reporting engine is a powerful, user-friendly tool for creating a wide array of reports and interactive dashboards. Once employee data is linked to your records, building these reports becomes straightforward:

#### Accessing the Reporting Module:

Navigate to **Reports > View / Run** in the ServiceNow navigator. From here, you can create new reports or access existing ones.

#### Creating a New Report:

1. Click the **Create New** button.

2. **Report Name:** Provide a descriptive name (e.g., "Incidents by Affected Department").

3. **Source Type:** Select 'Table'.

4. **Table:** Choose the table you have linked employee data to (e.g., 'Incident').

5. Click **Next**.

#### Configuring Report Data (Type and Filters):

1. **Report Type:** Select the desired visualization (e.g., Bar, Pie, List, Pivot Table). For analytical reports, Bar charts or Pivot Tables are often most effective.

2. **Group by:** This is where you leverage the linked employee data. For an 'Incidents by Affected Department' report, you would select the 'Affected Department' field. For 'Open Change Requests by Employee Location', you would select 'Affected Location' (assuming the Change Request table has been linked to the user requesting or impacted by the change).

3. **Aggregation:** Typically, you will want to count the records, so select 'Count'.

4. **Filter Conditions:** Apply filters to refine your data. For example, you might filter incidents to only include 'Active' or 'Resolved' states, or filter requests to a specific date range.

5. Click **Run** to see a preview of your report.

#### Adding Reports to Dashboards:

Once a report is created, it can be added to a dashboard for consolidated viewing and ongoing monitoring.

1. Navigate to **Self-Service > Dashboards**.

2. Click **New** to create a dashboard or open an existing one.

3. On the dashboard, click the **Add Widgets** icon (usually a '+' or similar). A widget library will appear.

4. Search for your newly created report by name. Drag and drop the report widget onto your dashboard.

5. Save or update the dashboard.

### Examples of Enhanced Reports

The ability to link records to employee attributes opens up a new dimension of reporting. Here are a few examples of the types of reports that become easily achievable:

#### 1. Incidents by Employee Department

* **Table:** Incident
* **Report Type:** Bar
* **Group by:** Affected Department
* **Aggregation:** Count
* **Conditions:** Incident State is not Closed
* **Value:** This report visualizes which departments are currently experiencing the most open incidents, helping to identify high-demand areas or departments facing significant IT challenges.

#### 2. Open Change Requests by Employee Location

* **Table:** Change Request
* **Report Type:** Pie Chart
* **Group by:** Affected Location (assuming linking is done for Change Requests)
* **Aggregation:** Count
* **Conditions:** State is not Closed, Implemented, or Cancelled
* **Value:** This report shows the distribution of ongoing change activities across different physical or logical locations, aiding in resource planning and impact assessment for specific sites.

#### 3. Service Catalog Requests by Employee Cost Center

* **Table:** Requested Item
* **Report Type:** Pivot Table
* **Rows:** Requested Item
* **Columns:** Affected Cost Center
* **Aggregation:** Count
* **Conditions:** Stage is not Closed Complete
* **Value:** This provides a detailed breakdown of service demand by cost center, offering financial insights into which business units are utilizing specific services most frequently.

#### 4. Incidents by Manager (for Escalation or Trend Analysis)

* **Table:** Incident
* **Report Type:** List
* **Columns to Display:** Number, Short Description, State, Affected User, Affected Manager, Opened
* **Conditions:** State is Resolved or Closed, Manager is not empty
* **Sorting:** Opened (Descending)
* **Value:** This report can be used to identify managers whose teams are frequently involved in resolved or closed incidents, potentially highlighting areas where additional training or support might be beneficial for IT users or for analyzing the impact of incidents on specific teams.

### Benefits of Direct Access to Employee Attributes

The advantages of having direct access to these employee attributes within the context of service records are manifold:

* **Time Savings:** Eliminates manual lookups, allowing IT staff to generate reports and analyze data much faster.
* **Improved Data Accuracy:** Reports are based on direct, validated links to the authoritative sys\_user table, reducing errors from manual data entry or outdated information.
* **Enhanced Decision Making:** Data-driven insights into departmental or locational trends empower better strategic decisions regarding IT resource allocation, service improvements, and support models.
* **Increased Efficiency:** Streamlines the process of understanding the "who" and "where" behind service events, leading to quicker diagnoses, more targeted support, and more effective problem resolution.
* **Empowered Reporting:** Non-technical users can more easily build and understand reports, as the contextual information is directly present in a familiar format.

By leveraging the Now Platform's reporting capabilities in conjunction with linked employee data, we move beyond simple record-keeping to intelligent, actionable analysis, driving greater efficiency and effectiveness across our IT operations.

## Testing and Validation

A critical phase in the successful deployment of any new functionality is rigorous testing and validation. This ensures that the implemented solution not only functions as designed but also meets the business requirements and provides accurate, reliable data. For the employee record linking project, our testing strategy is multi-faceted, encompassing unit testing of individual components, integration testing to verify data flow between systems, and user acceptance testing (UAT) to confirm real-world usability and stakeholder satisfaction. The objective is to validate the accuracy of the linked employee details and the seamless operation of the entire system.

### Unit Testing

Unit testing focuses on the smallest testable parts of the application, ensuring that each component works correctly in isolation. For this project, unit testing will primarily target:

* **Reference Fields:** We will verify that the newly created reference fields (e.g., 'Affected User', 'Affected Department', 'Affected Location', 'Affected Manager') on the target tables correctly reference the sys\_user table and its related attributes. This involves checking field properties, data types, and the integrity of the reference.
* **Scripts (Business Rules and Client Scripts):** Each script responsible for populating the employee detail fields will be tested individually.
  + **Display Business Rules:** We will test that when a user is selected in the 'Affected User' field, the business rule correctly queries the sys\_user table and populates the g\_scratchpad variables with the correct department, location, and manager information. Test cases will include scenarios with valid user IDs, non-existent user IDs, and when the 'Affected User' field is cleared.
  + **Client Scripts (onLoad/onChange):** We will test that the onLoad script correctly reads values from g\_scratchpad and populates the corresponding fields on the form. For onChange scripts, we will test that selecting or changing the 'Affected User' field correctly triggers the script to update the detail fields, either via g\_scratchpad or, if using GlideAjax, through the Script Include. Test scenarios will cover initial form loads, changes after load, and clearing the user field.
* **UI Policies/Attributes:** If UI Policies or dictionary attributes (like readonly) are used to manage field visibility or behavior, these will be tested to ensure they are applied correctly based on the defined conditions.

For each unit test, we will document the test case, the expected outcome, the actual outcome, and a pass/fail status. Any discrepancies will be logged as defects for correction.

### Integration Testing

Integration testing validates that different modules and components of the application work together seamlessly. In this project, integration testing will focus on the end-to-end flow of data from user selection to the population of employee details on the record.

* **User Selection to Field Population:** The primary integration test involves creating or modifying a record in a target table (e.g., an Incident). We will select an 'Affected User' from the sys\_user table using the auto-completion feature. The test will then verify that the 'Affected Department', 'Affected Location', and 'Affected Manager' fields are populated accurately and in near real-time.
* **Data Consistency Across Tables:** We will test the linking and population process across all implemented target tables (e.g., Incident, Requested Item, Change Request) to ensure consistency in behavior and data accuracy.
* **Interaction with Existing Functionality:** We will ensure that the new fields and scripts do not negatively impact existing ServiceNow functionalities, such as workflow approvals, notifications, or other business logic that relies on these records. This includes testing if existing reports or dashboards are affected.
* **Performance Testing:** While not exhaustive performance testing, we will conduct basic checks to ensure that selecting a user and updating fields does not cause significant delays or unresponsiveness in the ServiceNow interface. This is particularly important for the onChange client script and its GlideAjax calls.

Integration testing simulates real-world usage scenarios to ensure the entire solution operates as an integrated system.

### User Acceptance Testing (UAT)

User Acceptance Testing is the final stage of testing, where the intended end-users and key stakeholders validate the solution against business requirements. This phase is crucial for gaining confidence that the system will be effective in a live environment.

* **Test Scenarios:** A set of realistic test scenarios will be developed based on common operational tasks and reporting needs. These scenarios will guide the testers through the process of creating, updating, and analyzing records using the new functionality. Examples include:
  + "As a help desk agent, create a new incident, assign it to an affected user from the 'Sales' department, and verify that the 'Affected Department' field correctly shows 'Sales'."
  + "As an IT manager, run a report on open incidents grouped by 'Affected Location' and verify that the data reflects the actual distribution of incidents across office locations."
  + "As a service desk analyst, request a new laptop via the Service Catalog for a user in the 'Engineering' department, and confirm that the 'Affected Department' field on the resulting Request Item is populated correctly."
* **Testers:** A representative group of users, including Service Desk agents, IT support staff, and potentially business unit stakeholders who rely on ServiceNow reports, will participate in UAT.
* **Validation Process:** Testers will execute the predefined scenarios and provide feedback on usability, data accuracy, and overall functionality. They will validate that the linked employee details (Department, Location, Manager) are correctly displayed and that reports generated using these fields provide the expected insights.
* **Defect Management:** Any issues or deviations from expected behavior reported during UAT will be logged, prioritized, and addressed. Testers will re-test any fixes provided.

Successful completion of UAT signifies that the solution meets the users' needs and is ready for deployment.

### Data Validation

Throughout all testing phases, data validation is paramount. This involves:

* **Accuracy Checks:** Comparing the employee details populated in ServiceNow records against the source data in the sys\_user table and, where applicable, against HR system data to ensure accuracy.
* **Completeness Checks:** Verifying that all required employee attributes are consistently populated when a user is linked.
* **Referential Integrity:** Ensuring that the links to the sys\_user table and related attribute tables (like cmn\_department, cmn\_location) are valid and correctly maintained.

A comprehensive testing and validation strategy ensures that the employee record linking project delivers on its promise of enhanced reporting and administrative efficiency with high data integrity.

## User Training and Documentation

To ensure the successful adoption and effective utilization of the new employee record linking functionality within ServiceNow, a comprehensive training and documentation plan is essential. This plan outlines the approach for educating both end-users and administrators on how to leverage the enhanced capabilities, focusing on the simplified process of linking employee data and the resulting benefits for daily workflows and reporting. Continuous support and knowledge transfer are key components to guarantee long-term success and user proficiency.

### Training Plan for End-Users

End-users, such as help desk agents, service request fulfillers, and IT support staff, will be trained on how to interact with the enhanced forms and utilize the new functionality. The training will emphasize the practical benefits and ease of use:

* **Understanding the 'Affected User' Field:** Training will clearly explain the purpose of the new 'Affected User' reference field. Users will learn how to utilize the auto-completion feature to quickly search for and select the correct employee, making the process efficient and error-free.
* **Automatic Data Population:** A key highlight will be the automatic population of employee details such as Department, Location, and Manager once an 'Affected User' is selected. Users will be shown how these fields provide immediate context without requiring any manual data entry. This will be demonstrated through practical examples relevant to their daily tasks, such as creating incidents or fulfilling service requests.
* **Impact on Workflows:** The training will explain how this integration streamlines their workflows. For instance, faster identification of affected users and their departments can lead to quicker incident triage, more targeted communication, and improved resolution times. It reduces the need for cross-referencing information, saving valuable time.
* **Reporting Awareness:** While end-users might not directly build reports, they will be made aware that their accurate use of the 'Affected User' field directly contributes to more insightful departmental and locational reporting, highlighting the importance of their role in data accuracy.

Training will be delivered through a combination of methods, including interactive online modules, live virtual training sessions, and hands-on practice environments within a sub-production ServiceNow instance.

### Training Plan for Administrators and Support Staff

IT administrators and support personnel require a deeper understanding of the technical aspects and ongoing maintenance of the new functionality. Their training will cover:

* **Technical Overview:** A review of the implemented solution, including the configuration of reference fields, business rules, client scripts, and any related Script Includes. Understanding the technical underpinnings is crucial for troubleshooting and future enhancements.
* **Data Management and Integrity:** Training on the importance of maintaining the accuracy and completeness of the sys\_user table, as this directly impacts the data populated in linked records. This includes understanding data synchronization processes with HR systems.
* **Troubleshooting Common Issues:** Guidance on identifying and resolving common issues, such as incorrect data population, performance degradation, or errors in script execution. This will include reviewing logs and understanding how to debug issues related to the implemented components.
* **Reporting Capabilities:** A more in-depth session on how to leverage the linked employee data to build custom reports and dashboards, demonstrating the power of the reporting engine and how to extract meaningful insights for various stakeholders.
* **Change Management:** Procedures for managing any future changes or updates to the system, including testing methodologies and deployment protocols.
* **User Support:** How to effectively support end-users encountering questions or issues related to the new functionality.

Administrator training will be more technical, potentially including workshops and access to documentation detailing the architecture and configuration of the solution.

### User Guides and Documentation

Comprehensive documentation is vital for both immediate reference and long-term knowledge retention. The following documentation will be created:

* **End-User Quick Reference Guide:** A concise, visually driven document or job aid that quickly explains how to select an 'Affected User' and highlights the automatic population of details. It will focus on the "how-to" and the immediate benefits to their daily tasks.
* **Administrator's Guide:** A detailed technical document outlining the implementation steps, configurations, scripts, and architectural design of the employee record linking solution. This guide will serve as a critical resource for maintenance, troubleshooting, and future modifications. It will include details on the specific tables, fields, business rules, and client scripts involved.
* **Reporting Guide:** A guide for administrators and power users explaining how to build reports and dashboards using the linked employee data. It will include examples of common reports and instructions on using the ServiceNow reporting interface.
* **Knowledge Base Articles:** Short, searchable articles addressing frequently asked questions (FAQs) and specific use cases, accessible through the internal knowledge base.

All documentation will be stored in a central, accessible repository, such as the organization's intranet or ServiceNow's Knowledge Management module, ensuring easy access for all relevant personnel.

### Ongoing Support and Knowledge Transfer

To ensure sustained success, ongoing support mechanisms will be established:

* **Dedicated Support Channel:** A designated channel (e.g., a specific email alias, a Service Desk queue, or a Microsoft Teams channel) for users to ask questions or report issues related to the new functionality.
* **Regular Q&A Sessions:** Periodic Q&A sessions or "office hours" for end-users and administrators to address any lingering questions or concerns after the initial training.
* **Feedback Loop:** Establishing a mechanism for users to provide feedback on the functionality and documentation, allowing for continuous improvement and updates.
* **Mentorship and Cross-Training:** For administrators, fostering knowledge transfer through peer mentorship or cross-training sessions ensures that critical knowledge is distributed and not solely reliant on one individual.

By investing in thorough training and providing robust documentation and ongoing support, we empower our users and administrators to fully leverage the benefits of integrated employee data in ServiceNow, driving greater efficiency and improved reporting across the organization.

## Future Enhancements and Considerations

While the current implementation provides a solid foundation for linking ServiceNow records to employee data, several future enhancements and considerations can further maximize the value of this integration. These opportunities aim to deepen the contextual understanding, automate more complex workflows, and ensure the long-term scalability and maintainability of the solution. By looking ahead, we can continue to evolve our ServiceNow platform into an even more powerful tool for administrative efficiency and data-driven decision-making.

### Potential Future Enhancements

* **Linking to Other Employee-Related Tables:** Beyond the sys\_user table, opportunities exist to link records to other employee-centric tables. For example:
  + **HR Cases (**sn\_hr\_core\_case**):** Linking incidents or requests related to HR matters to the specific HR Case could provide a holistic view of an employee's interaction with HR and IT support.
  + **Learning and Development (LMS/Now Learning):** Connecting training records or requests to employee profiles could help analyze training effectiveness and identify skill gaps in relation to service adoption or incident trends.
  + **Onboarding/Offboarding Workflows:** Ensuring that relevant IT service requests or incident data are linked to employee onboarding or offboarding tasks can streamline these critical HR processes.
* **Automating Workflows Based on Employee Attributes:** The integrated employee data can be leveraged to trigger or modify existing workflows. Examples include:
  + **Assignment Rules:** Automatically assigning incidents or requests to specific support groups based on the affected user's department or location.
  + **Approvals:** Routing change requests or service catalog requests for approval to the affected user's manager, ensuring faster and more relevant approval cycles.
  + **Notifications:** Triggering targeted notifications to departmental managers or specific HR personnel when incidents or requests reach a certain severity or stage for users within their purview.
* **Leveraging Performance Analytics (PA):** For deeper, trend-based insights, Performance Analytics can be configured to track metrics related to employee attributes. This could include:
  + Tracking the average number of incidents per department over time.
  + Monitoring the resolution rate of requests by location.
  + Analyzing the impact of changes on different employee segments.
* PA can provide sophisticated trend analysis, forecasting, and target-setting capabilities built upon the directly linked employee data.
* **Extending to Other Modules:** The same linking methodology can be applied to other ServiceNow modules, such as Asset Management (linking assets to users), CMDB (linking CIs to the teams or departments that own or use them), or Project Management (linking projects to key stakeholders' departments).

### Key Considerations for Implementation and Maintenance

* **Data Synchronization:** Ensuring that the employee data in the sys\_user table remains accurate and up-to-date is critical. This often involves establishing robust data synchronization processes with authoritative sources, such as the organization's Human Resources Information System (HRIS). Any discrepancies or delays in synchronization can lead to inaccurate reporting and misinformed decision-making. Regular audits of data integrity are recommended.
* **Access Control (ACLs):** Carefully configure Access Control Lists (ACLs) to manage who can view, create, or modify the linked employee data fields. For instance, a standard end-user might only need to see their own department and manager, while an IT manager might require visibility into broader departmental data. Ensuring that sensitive employee information is protected according to privacy policies is paramount.
* **Performance Impacts:** While the current implementation is designed to be efficient, introducing new fields and scripts can have performance implications, especially on heavily used tables or during peak usage times. It is important to:
  + Optimize scripts for performance, using efficient GlideRecord queries and minimizing unnecessary operations.
  + Monitor system performance after implementation and regularly thereafter.
  + Consider indexing key fields that are frequently used in reporting or scripting logic.
  + For very large datasets or complex reporting needs, evaluate the use of Performance Analytics which is optimized for trend analysis and reporting.
* **Scalability:** As the organization grows or the usage of ServiceNow expands, the solution should remain scalable. This means ensuring that the approach taken can handle an increasing number of users, records, and data attributes without significant degradation in performance or stability.
* **User Adoption and Change Management:** Even with effective training, ongoing reinforcement and clear communication about the benefits of the linked data are essential for sustained user adoption.

By proactively addressing these future enhancements and considerations, we can ensure that our employee record linking solution remains a valuable and evolving asset within the ServiceNow platform.

## Conclusion

This project has successfully demonstrated the significant value of linking ServiceNow records directly to employee data within the sys\_user table. By integrating crucial employee attributes such as department, location, and manager into our core operational records, we have substantially enhanced our reporting capabilities and streamlined administrative workflows. The ability to gain immediate context regarding the organizational placement and roles of individuals associated with incidents, requests, and changes provides a richer understanding of service impact and user needs.

The practical benefits realized include a marked improvement in reporting accuracy and depth, allowing for more informed decision-making and targeted service improvements. Furthermore, the automation of data population has reduced manual effort, minimized errors, and freed up valuable time for IT staff, thereby increasing overall operational efficiency. This initiative underscores the relational power of the ServiceNow platform and its capacity to drive tangible business improvements.

The successful implementation of this project highlights the importance of well-structured and contextualized data. By leveraging ServiceNow's robust capabilities, we have transformed raw operational data into actionable intelligence, reinforcing the platform's role as a cornerstone for efficient service management and continuous improvement within the organization.