importing and securing data in ServiceNow

Team ID : LTVIP2025TMID28721

Team Size : 4

Team Leader : Ravanam Teja Lakshman

Team member : Reddy Dinesh Babu

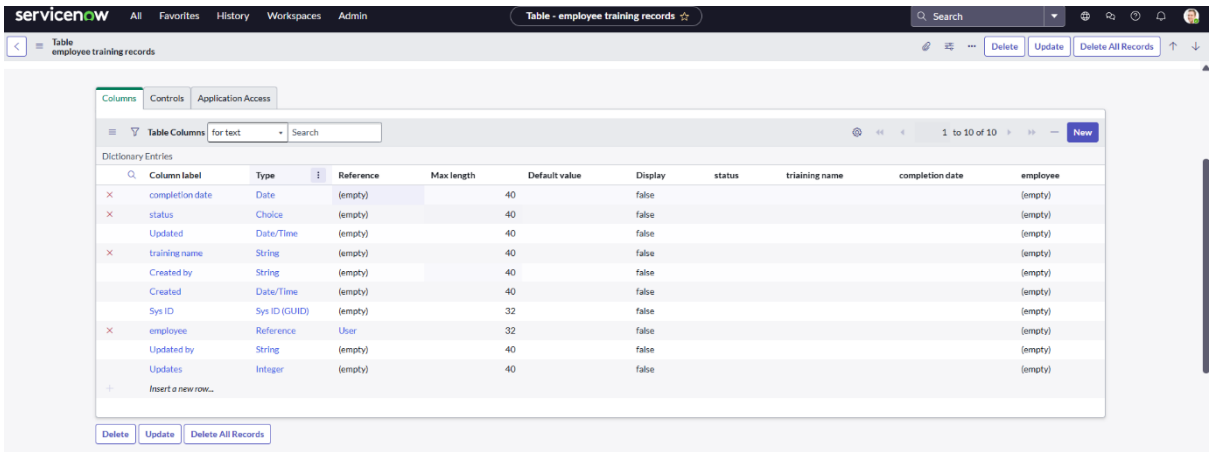
Team member : Rudrakshula Veera Venkata Satish

Team member : Sadhanala Likhith Varma

To solve the problem of *importing and securing data in ServiceNow* while *linking records to employees* and *pulling employee details* (like department) for easier reporting, here's a step-by-step guide that breaks down the process:

You want to:

1. Import records into ServiceNow.
2. Link each imported record to an employee.
3. Automatically populate employee details like *department* into the record.
4. Ensure data is *secured* (only the right users can view/edit it).
5. Make reporting easier using this linked data.



Step 1: *Prepare the Data Source*

Ensure your imported data (CSV, Excel, or external DB) includes a unique employee identifier (e.g., User ID, Email, or Employee Number) to match ServiceNow users.

C:\Users\likhi\OneDrive\Documents\Book.xlsx

Step 2: *Create the Target Table or Use an Existing One*

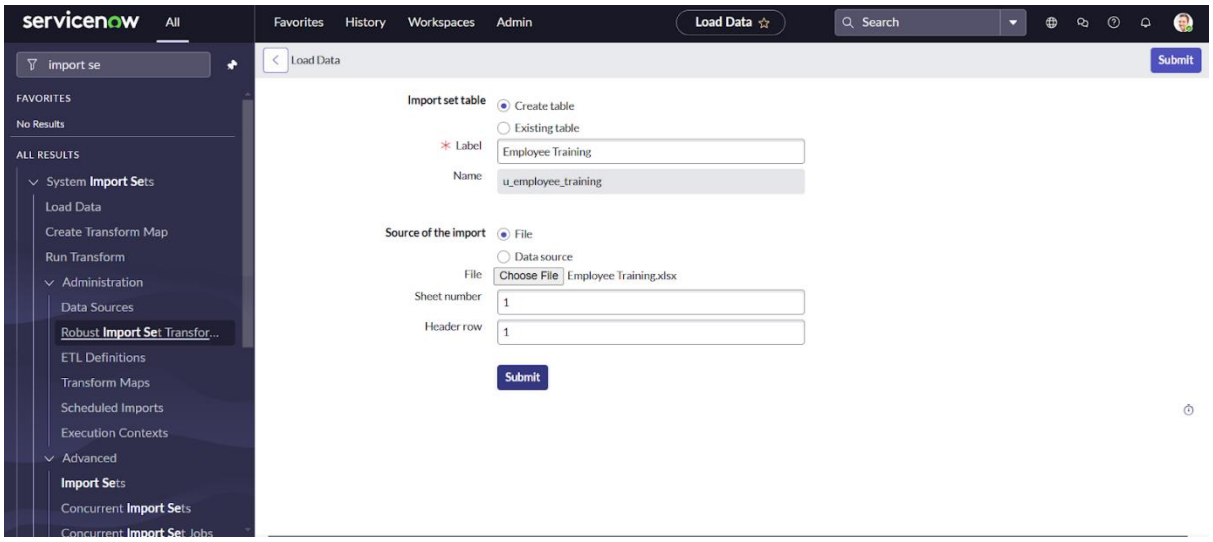
If you're importing into a custom table (e.g., u_custom_record):

* Navigate to *System Definition > Tables*

* Create a custom table if needed.

* Add a reference field: Employee (Reference to sys_user table)

* Add other fields, like Department (String or Reference to cmn_department)



Step 3: *Use Transform Maps to Import and Link Employee*

1. Go to *System Import Sets > Load Data*

2. Load your data source (CSV or external)

3. Create a *Transform Map*:

* Target Table: your custom table

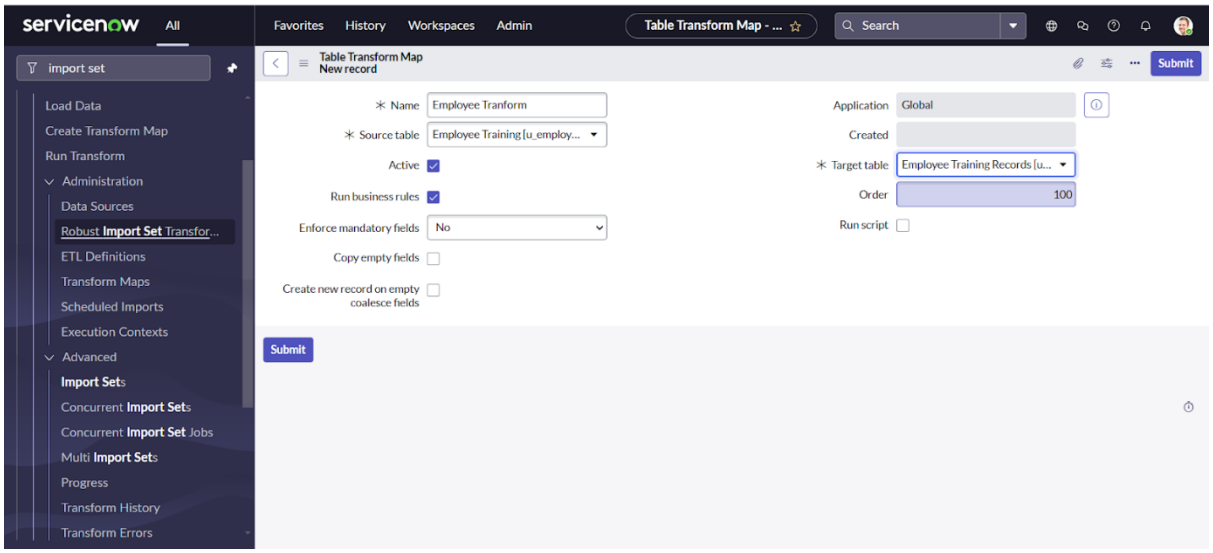
* Map fields like:

* Source field employee_email → Target field Employee (use a reference lookup to sys_user)

* Auto-populate Department using a *Scripted Field Map*:

javascript

```
// Scripted field map for Department field
var user = new GlideRecord('sys_user');
user.addQuery('email', source.employee_email);
user.query();
if (user.next()) {
  target.department = user.department.name; // or user.department for reference
}
```



4. Run Transform.

Step 4: *Auto-Populate Fields with Business Rules (Optional for Future Updates)*

If you want the department to stay in sync when the Employee changes:

* Create a *Business Rule* on your custom table
* Trigger on *Insert/Update*
* Script:

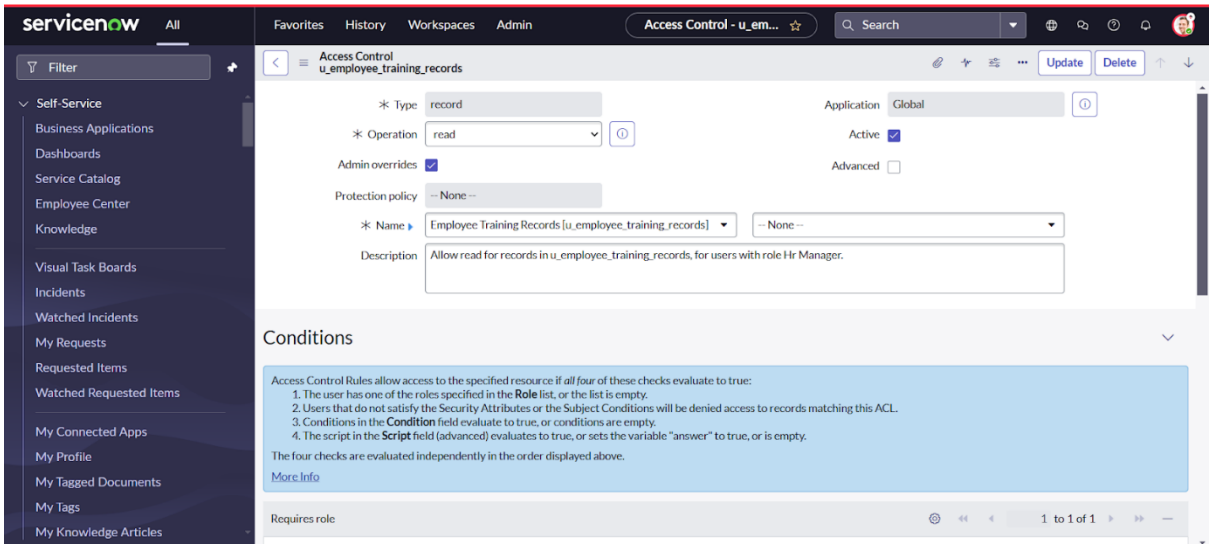
javascript

```
(function executeRule(current, previous /*null when async*/) {
  if (current.employee) {
    var user = new GlideRecord('sys_user');
    if (user.get(current.employee)) {
      current.department = user.department.name; // or user.department for reference field
    }
  }
})(current, previous);
```

Step 5: *Secure the Data*

Use *Access Control Rules (ACLs)*:

* Navigate to *System Security > Access Control (ACL)*
* Create ACLs on the table (e.g., u_custom_record):



* Record ACLs (read, write, delete)

* Field-level ACLs (hide department if needed)

* Conditions based on roles or ownership:

```javascript
gs.hasRole('admin') || current.employee == gs.getUserID();
```

Step 6: *Use Reporting/Performance Analytics*

Now that department is stored in the record:

* Use *Reports* to group by Department

* Use *Dashboards* to visualize employee-linked metrics

✅ Example Use Case Summary:

* *Imported Record*: Helpdesk survey, training completion, asset assignment
* *Employee Linked By*: Email or Employee ID
* *Pulled Info*: Department (from sys_user.department)
* *Secured By*: ACLs (only employee or manager can view)

Let me know if you'd like:

* A sample Transform Map
* Full script examples for ACLs
* A template for the Business Rule
* Help with setting up the data source or mapping fields