📄 Phase – 2: Backend Development & Configurations

Data Architecture

Description

The Data Architecture phase focuses on designing and configuring the backend data structure required to support the *Automated Network Request Management* solution in ServiceNow. This phase involves creating a custom table, defining fields with appropriate data types, and configuring forms to ensure structured data storage, consistency, and seamless integration with automation workflows.

A well-designed data architecture ensures scalability, accuracy, and efficient handling of network request data throughout the request lifecycle.

Activity 1: Creation of Table

Description

This activity involves creating a custom table in ServiceNow to centrally store and manage network-related request data. The table serves as the foundation for backend processing, automation logic, and reporting.

Implementation Steps

- Navigated to **System Definition > Tables** using the Application Navigator.
- Clicked **New** to create a new table.
- Entered the following table details:
    - **Label:** Network Database
    - **Name:** u_network_database (auto-generated by ServiceNow)
    - **Application:** Global

- **Extends table:** Not selected

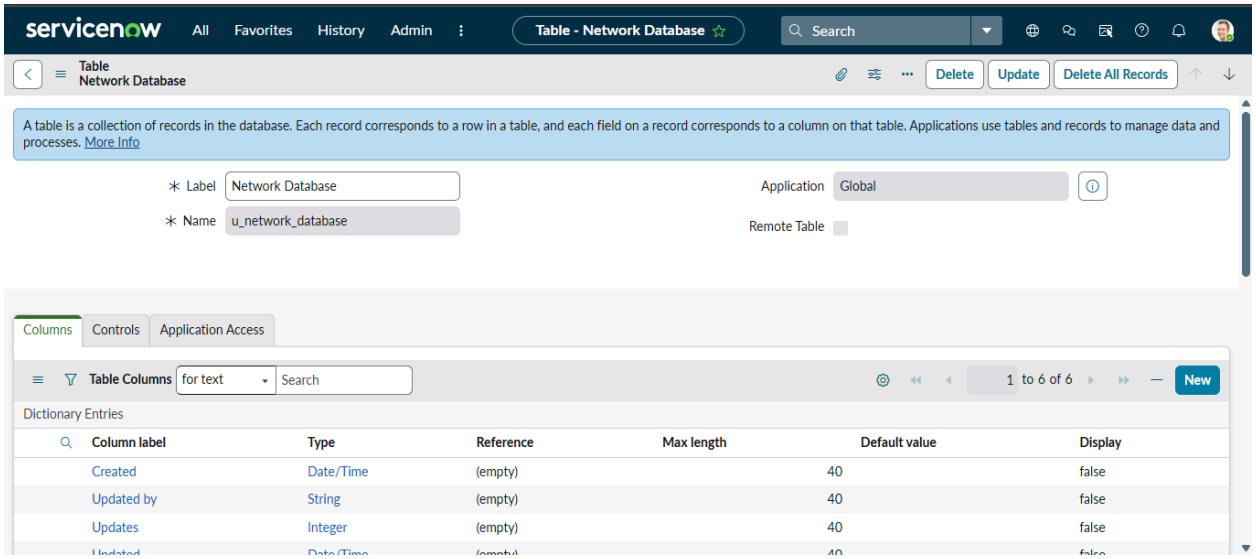
- Submitted the form to create the table.

In the Zurich release, schema fields such as `sys_id`, `sys_created_on`, and `sys_updated_on` are auto-generated by default, ensuring standard system metadata is available without additional configuration.

Outcome

A custom **Network Database** table was successfully created to store structured network request records in a centralized manner.

📸 **Screenshot:** *Network Database table creation screen*



Activity 2: Creation of Fields

Description

This activity focuses on adding custom fields (columns) to the Network Database table to capture essential data attributes required for network request processing.

Implementation Steps

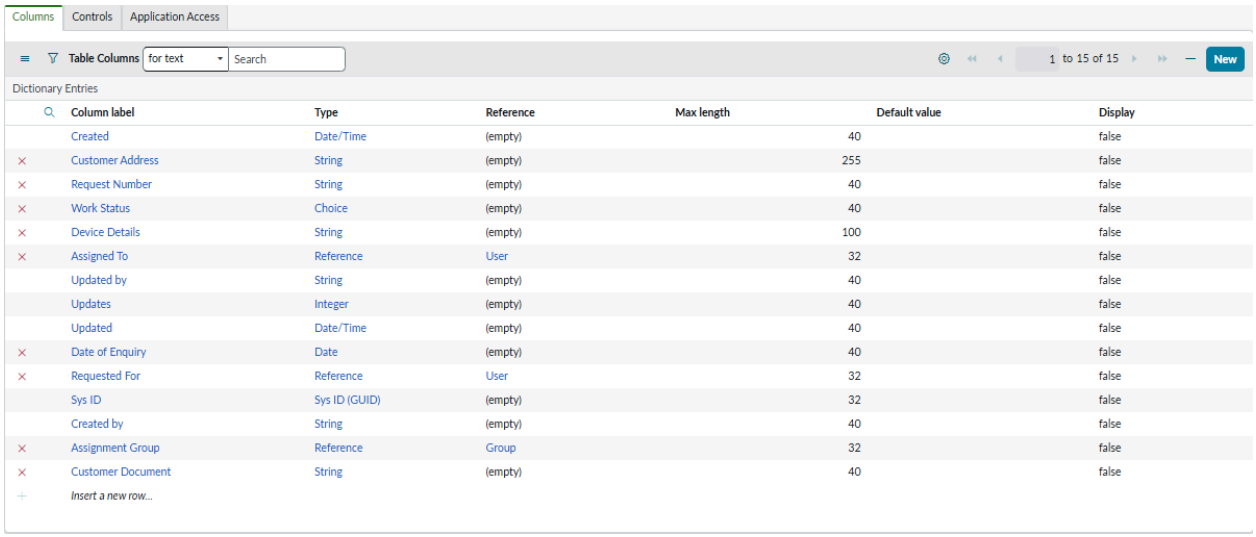
- Navigated to **System Definition > Tables**.
- Searched for and selected the **Network Database** table.
- Opened the **Columns** (Dictionary Entries) tab to view existing fields.
- Clicked **New** to create additional fields required for the solution.
- Created multiple fields to store request details such as request number, assignment information, customer data, device details, and request status.

Each field was created at the table level to ensure proper backend storage and integration with workflows and approvals.

Outcome

All required custom fields were successfully added to the Network Database table, enabling structured and comprehensive data capture for network requests.

📸 **Screenshot:** *Columns list showing newly created fields*



Column label	Type	Reference	Max length	Default value	Display
Created	Date/Time	(empty)	40		false
Customer Address	String	(empty)	255		false
Request Number	String	(empty)	40		false
Work Status	Choice	(empty)	40		false
Device Details	String	(empty)	100		false
Assigned To	Reference	User	32		false
Updated by	String	(empty)	40		false
Updates	Integer	(empty)	40		false
Updated	Date/Time	(empty)	40		false
Date of Enquiry	Date	(empty)	40		false
Requested For	Reference	User	32		false
Sys ID	Sys ID (GUID)	(empty)	32		false
Created by	String	(empty)	40		false
Assignment Group	Reference	Group	32		false
Customer Document	String	(empty)	40		false
Insert a new row...					

Activity 3: Define Field Properties

Description

This activity involves configuring field properties to control how data is stored, validated, and displayed across forms and records.

Field Configuration Details

- **Column Label:**
  Defines the user-facing name displayed on forms and lists (e.g., *Customer Address*).

- **Column Name:**
  Auto-generated internal name used by the system (e.g., `u_customer_address`).

- **Field Type:**
  Selected based on the nature of data being stored. Commonly used types include:
    - String
    - Integer
    - Choice
    - Reference
    - Boolean
    - Date
    - Date/Time

- **Max Length:**
  Defined for string fields to control data size where required.

- **Mandatory:**
  Enabled for critical fields to prevent incomplete data submission.

- **Default Value:**
  Configured where applicable to auto-populate values on record creation.

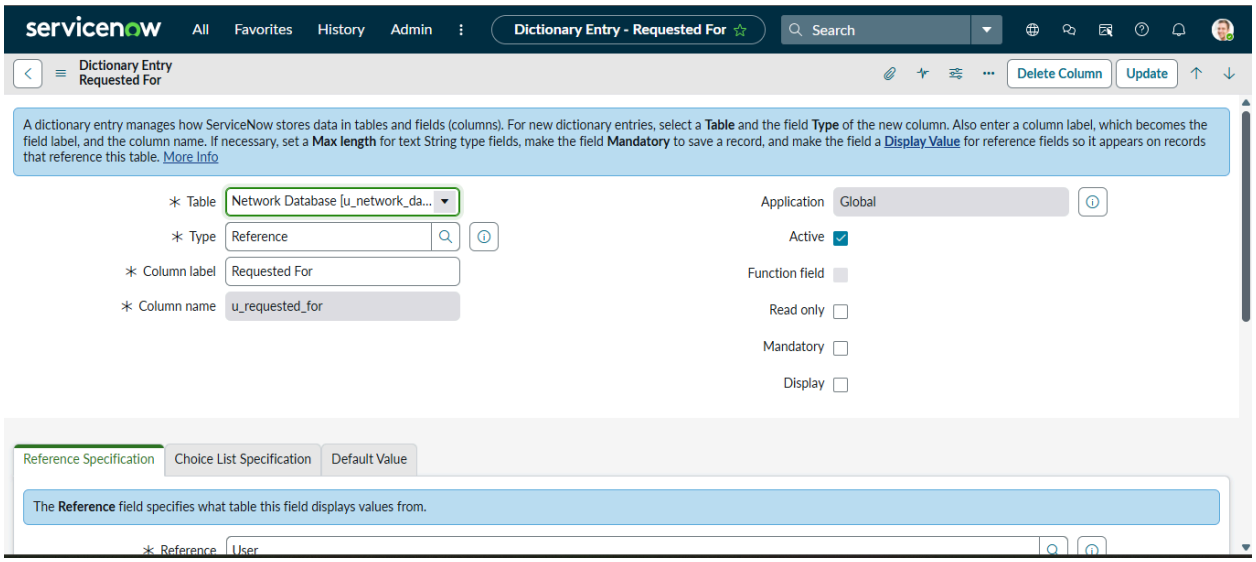
- **Read-Only:**
  Enabled for system-controlled fields to prevent manual modification.

- **Save Field:**
  Each field was saved using **Submit**, adding it to the table schema.

Outcome

Field properties were successfully configured to ensure data accuracy, validation, and consistency across all network request records.

📸 **Screenshot:** *Field properties configuration screen*



Activity 4: Add Fields to Form (Optional)

Description

This activity ensures that the newly created fields are available on the form for user interaction and record updates.

Implementation Steps

- Opened a record from the **Network Database** table.
- Selected **Configure > Form Layout** from the form menu.

- Moved required fields from the available list to the selected form layout.
- Saved the form configuration to apply changes.

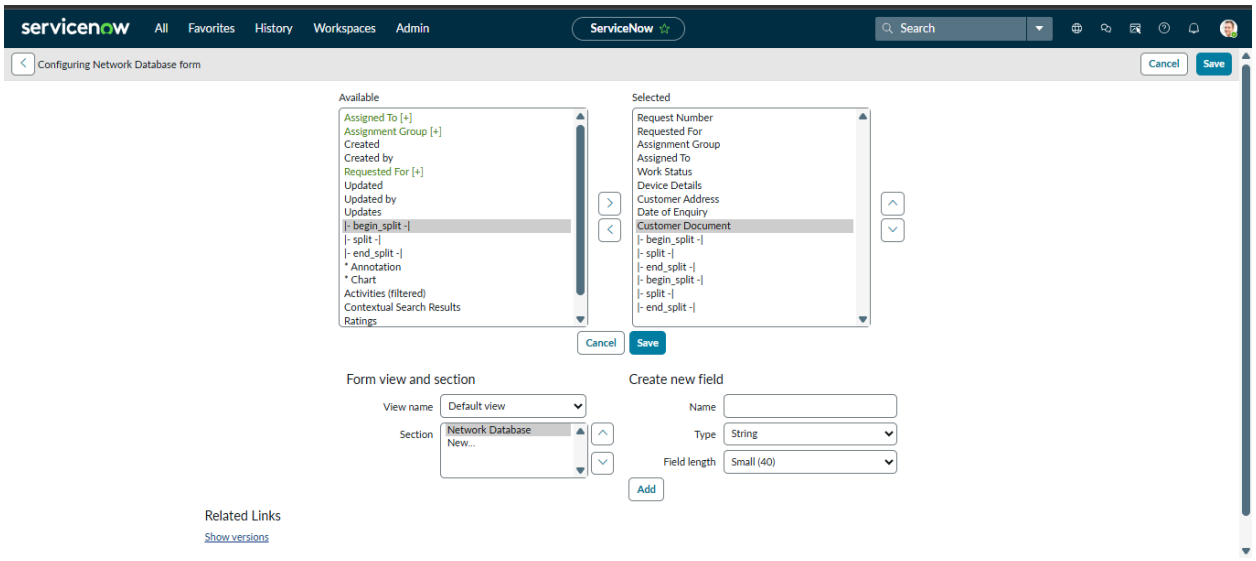
Testing

- Created a new record in the Network Database table.
- Verified that all fields appear correctly on the form.
- Validated mandatory fields, reference lookups, choice lists, and read-only behavior.

Outcome

Fields were successfully added to the form, enabling users and fulfilment teams to view and update network request data efficiently.

📸 **Screenshot:** *Network Database form with configured fields*



Conclusion

The Data Architecture phase successfully established a robust backend structure for the Automated Network Request Management solution. By creating a custom table, defining appropriate fields, and configuring form layouts, the solution ensures reliable data storage, improved data integrity, and seamless support for automation and approval workflows in subsequent phases.