DialogFlow CX Entities

Allen Sanders

AGENDA

1. Entities and Their Role in Extracting Data from User Inputs
2. Types of Entities in DialogFlow CX
3. Steps to Create Custom Entities
4. Configuring Entity Synonyms and Reference Values

01

Extracting Data Using Entities

Entities - Definition

Entities in Dialogflow CX

- Entities are used to extract structured data from user inputs
- They help interpret user phrases by recognizing specific words or phrases that match predefined categories

Entities - Definition

Definition and Purpose of Entities

DialogFlow CX entities are used to extract specific pieces of information from user inputs

Entities help recognize and classify data within user messages

Properly defined entities enhance the accuracy of the chatbot's responses

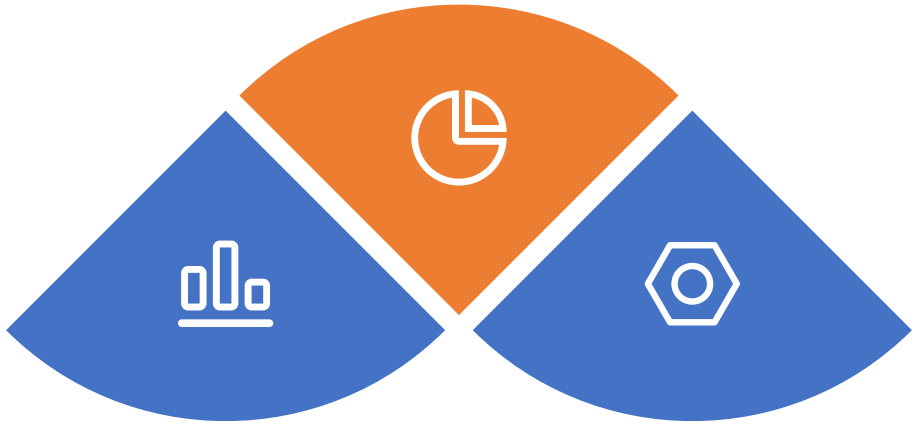
Role in DialogFlow CX

Supporting Intent Fulfillment

Support intent fulfillment by providing parameter values needed for business logic

Enabling System Understanding

Enable the system to understand and process user input by mapping it to structured data



Improving Capabilities

Improve chatbot and virtual agent capabilities by identifying key terms relevant to a given context

Implementing Entities in DialogFlow CX

Steps for Defining Entities

Identify key pieces of information that need to be extracted

Choose between system and custom entities based on requirements

Configure entity parameters and map them to specific user inputs

Entity Extraction in Conversations

Real-time Data Processing

Entities extract data as users interact with the chatbot

Extracted data can be used to tailor responses dynamically

Ensures user input is understood and processed accurately

Designing Effective Entities

Ensure entities are logically defined and consistently used

Regularly update and refine entities based on user feedback and analytics

Testing entities helps maintain high accuracy in data extraction

Consistency and Accuracy

Testing & Optimization

Continuous Improvement

Monitor entity performance using built- in DialogFlow CX tools

Optimize entities based on real-world usage and error rates

Iterate on entity definitions to improve bot functionality and user satisfaction

02

Types of Entities

System Entities

Predefined by DialogFlow CX

Automatically detect common data types such as numbers, dates, emails, and locations

Examples:

@sys.number (Extracts numerical values)

@sys.date (Recognizes dates like "tomorrow" or "March 15")

@sys.email (Identifies email addresses)

Custom Entities

Defined by developers to match specific words or phrases relevant to their application

Can include multiple variants for improved recognition

Examples:
@product_name with values: "iPhone", "Samsung Galaxy", "Pixel“

@operating_system with values: “Windows”, “Linux”, “MacOS”

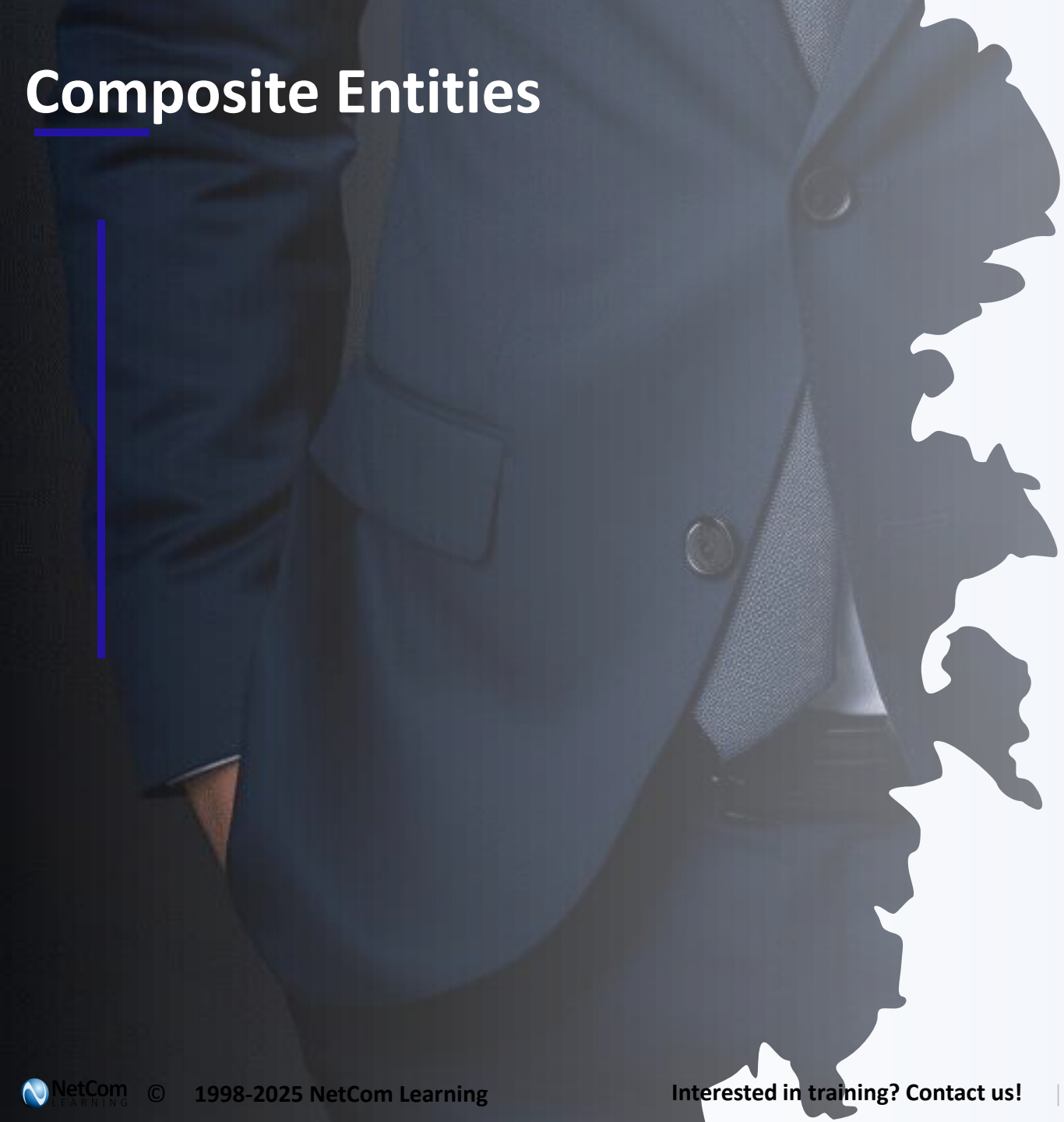
RegExp Entities

> Use regular expressions to define flexible matching patterns

Best for structured text inputs like order IDs (ORD-\d{5})

Examples:
@order_id with regex: ORD-\d{5} (matches "ORD-12345")

Composite Entities



Used to group multiple entities into a hierarchical structure

Examples:

Booking entity with attributes:

@location (e.g., "Paris")

@date (e.g., "April 10")

List Entities

A specific type of custom entity where predefined lists are used to recognize variations

Supports synonyms and reference values

03

Creating Custom Entities

Benefits of Using Custom Entities

Improving Accuracy

Custom entities enhance the chatbot's accuracy in recognizing and processing user inputs, leading to more precise and effective responses

Enhancing User Experience

By accurately recognizing user inputs, custom entities provide a more seamless and satisfying interaction experience for users

Accessing the Dialogflow CX Console

01

Login and Project Selection

Access DialogFlow CX Console, select the appropriate Google Cloud project containing the desired DialogFlow CX agent

02

Navigating to Entities Section

In the console's left sidebar, click on Entities and then click on the "Create Entity" button to start the entity creation process

Defining the Entity

Naming the Entity

Enter a descriptive name for the entity, ensuring it clearly identifies the type of data it will represent (e.g., @car_model, @payment_method)

Choosing Entity Type

Select between List, RegExp, and Composite types based on the structured data needed, with options for fuzzy matching to handle typos

Adding Entity Values

Input canonical values and synonyms to account for various ways users might phrase input and enable automated expansion if necessary

Adding Entity Values

1. Under **Entity Values**, enter **canonical values** (standardized forms of the entity).
2. Add **Synonyms** for each value (alternative ways users might phrase the value).
3. Enable **allow automated expansion** if users might enter values not explicitly defined.

Example: Custom Entity for Car Models (`@car_model`)

Canonical Value	Synonyms
`Toyota Corolla`	`"Corolla"`, `"Toyota Corolla"`, `"Corolla 2023"`
`Honda Civic`	`"Civic"`, `"Honda Civic"`, `"Civic 2022"`
`Ford Mustang`	`"Mustang"`, `"Ford Mustang"`, `"Mustang GT"`

Implementing Custom Entities in Intents

Example: Car Rental Intent

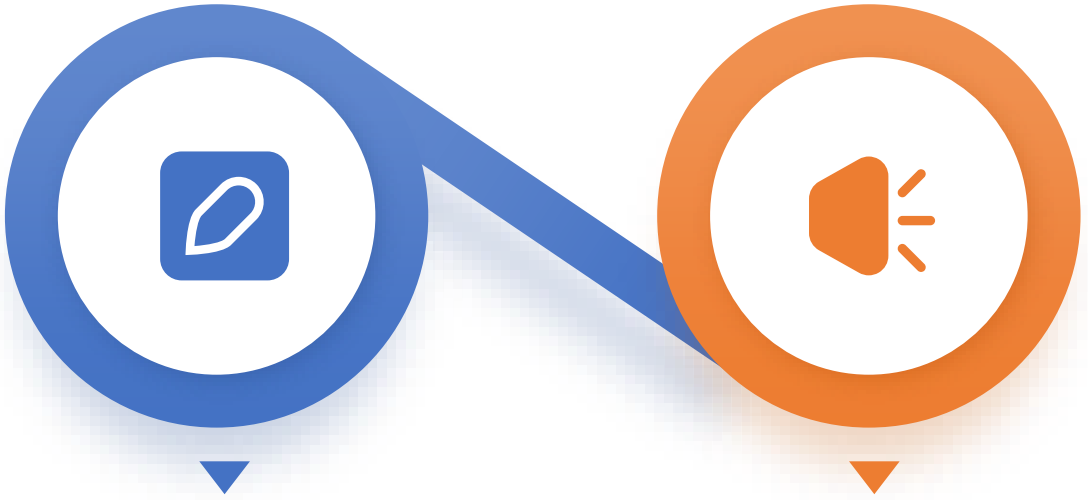
Defining the Intent

Create an intent (e.g., Booking a Rental Car) and map it to custom entities to extract and handle specific user inputs like car models

Mapping Parameters

Map the extracted parameters to entity types, ensuring required fields are marked and including prompts for additional information if needed

Using RegExp Entities



When and How to Use

Use RegExp Entities for structured patterns like order IDs, enabling precise recognition with regular expressions

Example: Order ID Recognition

Create a custom entity using a regular expression (e.g., ORD-\d{5}) to recognize structured inputs like order IDs

Grouping Multiple Entities

Definition and Use Case

Composite entities group multiple entities to provide structured information, useful for complex inputs requiring detailed parameter extraction

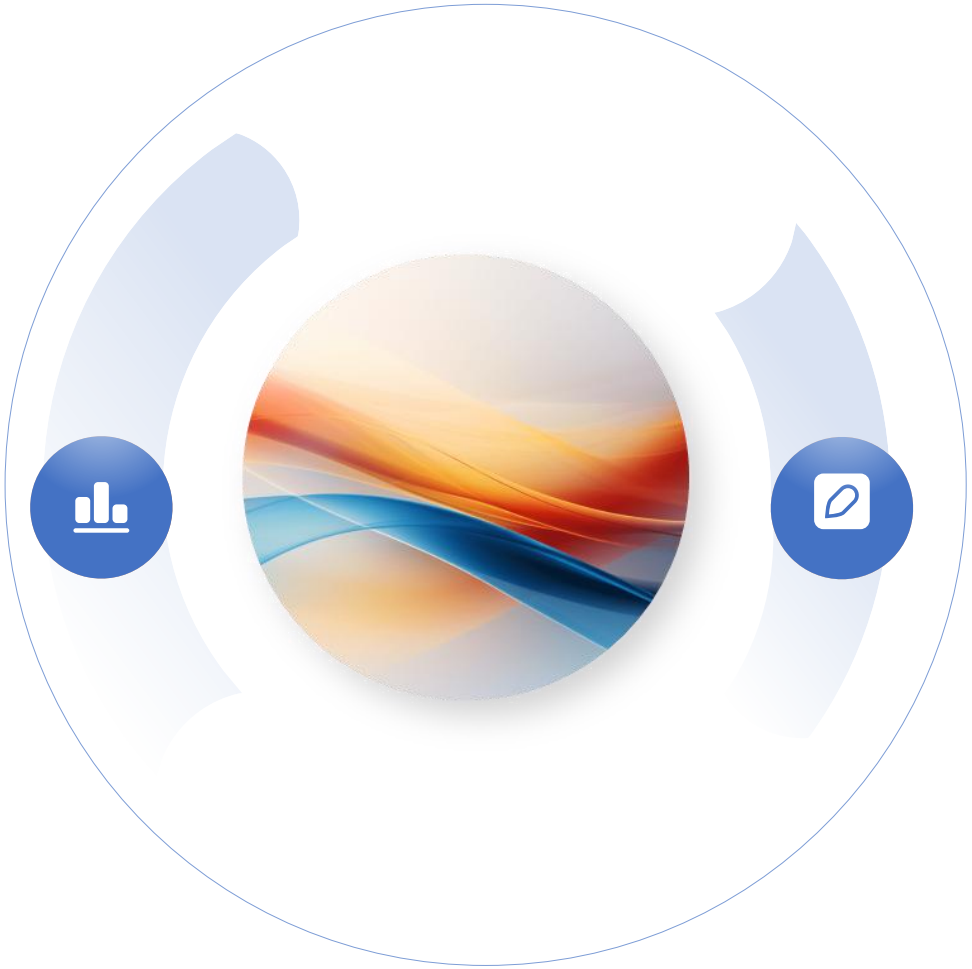
Example: Hotel Booking

Create a composite entity for hotel bookings, combining sub-entities like hotel names, locations, and dates for check-in and check-out

Testing Using the Simulator

Entering User Inputs

Use the simulator to input various user phrases and verify if the correct entity is recognized and extracted accurately



Debug Logs

Enable debug logs to monitor entity recognition and adjust synonyms or fuzzy matching settings as needed for better accuracy

Testing Using the API

Programmatic Verification

- You can call the DialogFlow CX API to test entity recognition programmatically
- Can be used to help ensure the custom entities perform as expected in real scenarios

Example – Intent: Booking a Rental Car

User Input:

> "I want to rent a Tesla Model S."

Custom Entity for Car Models (`@car_model`)

- Extracted Parameter: `"Tesla Model S"`
- Synonyms: `"Model S"`, `"Tesla S"`, `"S P100D"`

Intent Parameter Mapping

- Parameter Name: `carModel`
- Entity Type: `@car_model`
- Required: ✅
- Prompt: `"Which car model would you like to rent?"`

Example – Composite Entities

- If multiple entities need to be grouped together for more structured information.

Example: Custom Entity for Hotel Booking (`@hotel_booking`)

Composite Entity	Sub-Entities
`Hotel Name`	`@hotel_name`
`Location`	`@city`
`Check-in Date`	`@sys.date`
`Check-out Date`	`@sys.date`

User Input:

> "I want to book a room at Hilton in New York from April 10 to April 15."

Extracted Parameters

- `@hotel_name = "Hilton"`
- `@city = "New York"`
- `@sys.date (check-in) = "April 10"`
- `@sys.date (check-out) = "April 15"`

04

Entity Synonyms & Reference Values

Entity Synonyms

STEP. 01

What Are Entity Synonyms?

- Allow multiple variations of an entity value to be mapped to a single reference value
- Helps ensure that users can input data in different ways while maintaining a unified representation in the backend

STEP. 02

How Synonyms Work in DialogFlow CX

- When an entity with synonyms is detected in a user utterance, DialogFlow maps it to a pre-configured reference value
- Help improve natural language understanding (NLU) by allowing flexibility in user input
- The system does not store synonyms separately but uses them to normalize the extracted entity value and to train

STEP .03

Example 1: Recognizing Different Ways to Refer to the Same Product

Custom Entity
@smartphone_brand
User Input & Extracted Value

STEP. 04

Example 2: Handling Multiple Ways to Say a Location

Custom Entity @city
User Input & Extracted Value

STEP. 05

How to Configure Synonyms in DialogFlow CX

- Navigate to the DialogFlow CX Console
- Select Your Agent, then go to the Entities section
- Create or Edit an Entity
- Add an Entity Value (Canonical Form)
- Enter Synonyms for Each Value
- Enable Fuzzy Matching (optional, to allow approximate matching)
- Save and Test the Entity Recognition in the Simulator

Examples - Synonyms

Example 1: Recognizing Different Ways to Refer to the Same Product

Custom Entity: `@smartphone_brand`

Entity Value (Canonical)	Synonyms
`Apple`	`"iPhone"`, `"iOS"`, `"Mac phone"`
`Samsung`	`"Galaxy"`, `"Samsung phone"`, `"Note"`
`Google`	`"Pixel"`, `"Google phone"`

User Input & Extracted Value

- User: `"I want to buy a Galaxy."`
- Recognized entity: `@smartphone_brand = "Samsung"`

Example 2: Handling Multiple Ways to Say a Location

Custom Entity: `@city`

Entity Value (Canonical)	Synonyms
`New York`	`"NYC"`, `"New York City"`, `"Big Apple"`
`San Francisco`	`"SF"`, `"San Fran"`, `"Bay Area"`

User Input & Extracted Value

- User: `"Book me a flight to NYC."`
- Recognized entity: `@city = "New York"`

Reference Values

What Are Reference Values?

- Standardized form of an entity value used in backend processing
- While synonyms allow different user inputs, the reference value ensures that all variations are normalized into a single standard format

How Reference Values Work

- When a synonym is detected in a user's input, Dialogflow CX maps it to a reference value
- The reference value is what gets passed to fulfillment, webhook, or backend APIs

Example 1: Normalizing Car Models for Backend Processing

Custom Entity @car_model
User Input & Reference Value

Example 2: Handling Variations in Payment Methods

Custom Entity @payment_method
User Input & Reference Value

Examples – Reference Values

Example 1: Normalizing Car Models for Backend Processing

Custom Entity: `@car_model`

Reference Value (Canonical)	Synonyms
`Toyota Corolla`	`"Corolla"`, `"Toyota Corolla"`, `"Corolla 2023"`
`Honda Civic`	`"Civic"`, `"Honda Civic"`, `"Civic 2022"`
`Ford Mustang`	`"Mustang"`, `"Ford Mustang"`, `"Mustang GT"`

User Input & Reference Value

- User: `"I want a Mustang GT."`
- Extracted Value: `"Ford Mustang"`

Example 2: Handling Variations in Payment Methods

Custom Entity: `@payment_method`

Reference Value (Canonical)	Synonyms
`Credit Card`	`"Visa"`, `"MasterCard"`, `"Amex"`, `"credit"`
`PayPal`	`"PayPal"`, `"Paypal account"`, `"PayPal balance"`
`Cryptocurrency`	`"Bitcoin"`, `"Ethereum"`, `"crypto"`

User Input & Reference Value

- User: `"I'll pay with Bitcoin."`
- Extracted Value: `"Cryptocurrency"`

Enabling Fuzzy Matching

What Is Fuzzy Matching?

- Allows the system to recognize entity values even if they are misspelled or slightly different from the predefined values
- Useful for handling typos, pluralization, abbreviations, or phonetic variations

Example: Handling Typos in User Input

Custom Entity @fruit
User Input & Extracted Value

How to Enable Fuzzy Matching in Dialogflow CX

- Navigate to the Entity Page
- Open the Specific Entity
- Check the “Enable Fuzzy Matching” Option
- Save and Test the Entity Recognition in the Simulator

Examples – Fuzzy Matching

Example: Handling Typos in User Input

Custom Entity: `@fruit`

Reference Value (Canonical)	Synonyms
`Strawberry`	`"strawbery"`, `"strawberri"`, `"strawbry"`
`Blueberry`	`"blue berry"`, `"bluebry"`, `"bluberry"`
`Raspberry`	`"raspbery"`, `"rasberri"`, `"rasbry"`

User Input & Extracted Value

- User: `"I want a strawbery smoothie."`
- Recognized entity: `"Strawberry"` (due to fuzzy matching).

Testing & Validating Entity Synonyms & Reference Values

How to Test Entity Recognition

Use the Simulator in Dialogflow CX

Enable Logging & Analytics

- Monitor how users input entity values
- Update synonyms as needed based on real-world usage.

Use Webhooks for Custom Validation

If additional verification is needed, use webhooks to confirm entity values

Use Cases

1. E-commerce Chatbot

- User Input: `"Do you have sneakers?"`
- Extracted Entity: `@product_category = "Shoes"` (mapped via synonyms like `"sneakers"`, `"footwear"`, `"kicks"`).
- Backend Query: `"SELECT * FROM inventory WHERE category = 'Shoes'"`

2. Food Ordering Assistant

- User Input: `"I want a Pepsi."`
- Extracted Entity: `@drink = "Coca-Cola"` (if configured to map all sodas to a general category).
- Order Processing: `"Drink preference: Coca-Cola"`

3. Travel Booking Bot

- User Input: `"Book a ticket to LA."`
- Extracted Entity: `@city = "Los Angeles"` (mapped via synonyms like `"LA"`, `"L.A."`, `"City of Angels"`).
- API Call: `"Booking a flight to Los Angeles"`

Best Practices for Configuring Synonyms & Reference Values

✓ **Use Descriptive Reference Values** – Ensure backend systems receive standardized and meaningful data

✓ **Include Common Variations as Synonyms** – Think of different ways users might phrase the same entity

✓ **Enable Fuzzy Matching Where Needed** – Helps with typos and misspellings

✓ **Regularly Update Synonyms Based on Real User Input** – Use logs to refine entity values

✓ **Test Extensively in Different Scenarios** – Ensure correct entity mapping with diverse inputs

Thank you

NetCom LEARNING