Guidewire important Questions with Answer Part-2:

**1. What is retireable and versionable entity?**

**Retireable:**

A Retireable entity is an entity that can be logically "retired" instead of being

permanently deleted from the database. This helps maintain historical data while

preventing the entity from being used in future transactions.

**Example:**

Policy Type, Coverage Type, and User entities are often Retireable because even when they are no longer in active use, they might still be needed for historical claims.

**Versionable:**

A Versionable entity is an entity that maintains multiple versions of itself over time. Each time a record is updated, a new version is created, allowing the system to track changes over time.

**Example:**

* **Policy Period**: Tracks different versions of a policy over multiple renewal periods.
* **Coverage**: Maintains historical changes in coverage terms.
* **Claim**: Tracks modifications to claims throughout the claim lifecycle.

**2. What is Extendable, Exportable and Final in entity creation?**

**Extendable:**

An Extendable in entity is an entity that can be customized by adding new fields, new subtypes, or even new relationships. These entities allow insurers to tailor Guidewire applications to meet their business needs.

**Exportable:**

An Exportable is true it **must be serialized** so that it can be sent to an **external system** through web services.

**Final:**

A Final in entity is an entity that cannot be extended or modified. These entities are locked to prevent changes that could impact on core system functionality. We cannot be able to create subtypes.

**3. What are the four levels of validations?**

| **Validation Level Name** | **Code** | **Description** |
| --- | --- | --- |
| Load and save | load save | Claims and exposures imported from an external system must contain a minimal level of information to be saved in Claim Center. However, the system needs more information before an adjuster can work on them. |
| New loss completion | new loss | If you create a claim from the wizard, this level defines the minimum amount of information for it to be saved as a claim. |
| Ability to pay | payment | This level ensures that a claim has all the required data needed to make a payment on it. |
| Valid for ISO | iso | (Optional) This level verifies that all required fields are complete before sending to ISO. |
| Send to external system | external | (Optional) This level can verify if the claim has enough information before it is sent to an external system. In the base configuration, there is no functionality associated with this level. |

**4. What are the types of validation and level of validation?**

Validation is a general application behavior that helps you avoid making mistakes and avoid

saving invalid business data. Claim Center validates data in the following ways:

1. **Validation by Field-level–** Field-level validation is the process of validating individual

fields within an entity to ensure that each field contains valid, accurate, and acceptable data

before it can be processed or saved in the database, which can be implemented at:

* + Data model level – Includes data types and field validators.
  + User interface level by using validation expressions – Includes validation behavior tied to one or more specific fields, which can be implemented at the user interface level in Gosu code.

**Errors:**

Error is the required field, and it will not allow to go to next page or it will not allow to update the data in database util the error was cleared. For example,

such as "Policy Start Date," is left empty. The system throws an error preventing form submission.

**Warnings:**

Warning is the type of validation, and it will allow to next page or it will allow further steps to update the data in database even we did not compete the action. For example,

An entered phone number has an unusual format but is still acceptable. A warning is displayed.

**Validation:**

* 1. **validation by fields:**
* entity.reject()
* entity.rejectField(setRelativePath,errorLevel,errorMessage,warningLevel,warningMessage)
* entity.rejectSubField(relatdObject,setRelativePath,errorLevel,errorMessage,warningLevel,warningMessage)

**1.entity. reject** – This is used to **reject the entire entity** not just a specific field. This method triggers a validation error for the whole entity, stopping the process (e.g., saving or submitting the form) until the issue is resolved.

Syntax like, entity**. reject(null,null,null,null).**

For example,

If a policy’s expiration date is earlier than the effective date, the whole policy is invalid, so entity. reject would be used.

**2.entity. fieldReject** – This is used to **reject a specific field** within the entity. It triggers an error on that field rather than on the whole entity.

Syntax like, **entity. fieldReject(fieldpath, null,null,null,null).**

For example,

If a user enters an invalid value in the "Effective Date" field, you can use fieldReject to reject just that field.

**3.entity. subfieldReject** – This is used when the validation error is related to a **subfield** of a complex field or nested object.

Syntax like, **entity. subfieldReject(relatedpath, fieldpath, null, null, null, null).**

For example,

If the "Address" field in a policy has subfields like "ZIP Code" that are invalid, you can reject just the "ZIP Code" field within the address.

1. **Validation by Rules** – By defining rules, you can configure ClaimCenter to verify the maturity of a claim or exposure. You can also use rules to execute validation behavior at a global level when the error might not relate to one specific field. For example, an insurer allows up to five vehicles to be covered on a single personal auto policy. The underwriter enters six automobiles. The business data is invalid, but there is not any one field that is causing the error.

**5. What is workflow and how to create it?**

**Workflow** is a powerful feature that automates and manages business processes related to

claims handling. It allows you to define a sequence of steps (tasks, decisions, or actions) that

need to be executed in a specific order to process a claim efficiently. Workflows ensure

consistency, reduce manual effort, and improve the overall efficiency of claims handling.

**6. What is the use of authority limit?**

Authority Limit is a critical concept used to enforce segregation of duties and control

decision-making authority within the claims handling process. It ensures that claims are

handled by the appropriate personnel based on their role, experience, and authorization level.

This helps maintain compliance, reduce risk, and ensure that claims are processed efficiently

and accurately.

For example,

* A junior adjuster might have an authority limit of **$5,000**.
* A manager might have an authority limit of **$50,000**.

**How it works:**

1. Authority limits are assigned to user roles (e.g., Adjuster: $5,000, Manager: $50,000).
2. When a user performs an action (e.g., approving a reserve or making a payment), Claim Center checks their authority limit.
3. If within the limit, the action is allowed.
4. If above the limit, the action is escalated to a higher authority (e.g., supervisor or claims manager).

**7. What is eroding in check?**

When a check is issued in Claim Center, it typically represents a payment to a claimant,

vendor, or other party. The check amount is the total value of the payment. However, as

payments are made or expenses are applied, the available balance of the check is reduced.

This reduction is called eroding**.**

**How Eroding Works:**

1. **Check Issuance:**
   * A check is issued for a specific amount (e.g., $10,000).
   * Initially, the eroded amount is 0, and the available balance is 10,000.
2. **Applying Payments or Expenses:**
   * As payments or expenses are applied to the check, the eroded amount increases.
   * **Example:**
     + Payment 1: 2,000→ErodedAmount=2,000 → Available Balance = $8,000.
     + Payment 2: 1,000→ErodedAmount=3,000 → Available Balance = $7,000.
3. **Check Exhaustion:**
   * When the eroded amount equals the check amount, the check is fully exhausted, and the available balance becomes $0.
   * **Example:**

* Payment 3: 7,000→ErodedAmount=10,000 → Available Balance = $0

**8. How to create Activity Pattern?**

An Activity Pattern is a predefined template used to standardize the creation of activities. It allows organizations to define reusable activity structures, ensuring consistency and efficiency in task management.

We can create activity pattern in two ways in Guidewire Claim Center,

1. Manually Activity Pattern Creation

2.Automatically Activity Pattern Creation

**Manually Activity Pattern Creation:**

* Activities are created directly by a user within Claim Center.
* Created by users manually selecting an activity pattern.
* The user can generate Activities manually from Claim Center user interface.

Open claim 🡪 click on Actions 🡪 New Activity

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Automatic Activity Pattern Creation:**

* Activities are created automatically by the system based on predefined rules or triggers.
* Here we click automated only, in that we give the code. We can use the code by accessing this automatic activity pattern in the claim creation.

A screenshot of a computer

Description automatically generated

**9. Write a code step to create activity when claim loss cause is changed:**

**Step 1: Define the Activity Pattern:**

1. Navigate to Activity Patterns:
   * Log in to Guidewire Claim Center.
   * Go to the Administration tab.
   * Under Business Settings, click on Activity Patterns.
2. Create a New Activity Pattern:
   * Click New Activity Pattern.
3. Set the Details:
   * Code: Provide a unique code, e.g., "LOSS\_CAUSE\_CHANGE".
   * Category: Choose a suitable category like "General" or "ISO”.
   * Subject: Enter a default subject, e.g., "Loss Cause Changed".
   * Description: Add a placeholder for the description, e.g., "The Loss Cause has been updated."
   * Priority: Set the default priority, e.g., Normal.
   * Class: Select the class, e.g., “Task” or “Event”.
   * Callender Importance: Choose the Callender Importance, e.g., “Medium”.
4. Save the Activity Pattern:
   * Click Save.

**Step 2: Create the Rule to Use the Activity Pattern:**

**A white background with black and blue text

AI-generated content may be incorrect.**

**10. Create history when check is created:**

**Steps to Create History for a Check**

1. Identify the Trigger:
   * Use the Check creation event to trigger the rule.
   * In Claim Center, this is typically managed in the Check Rules ruleset.
2. Define the Rule:
   * Write a rule in the Check Rules section to add a history entry.

**A text on a white background

AI-generated content may be incorrect.**

**8. What is Event fire rule and uses?**

An Event Fire Rule is a mechanism that automatically triggers specific actions

when certain events occur in the system, such as creating activities, sending notifications, or

starting workflows. They help automate business processes, ensure consistency, and improve

efficiency in response to system changes.

**Use of Event Fire Rules:**

Event Fire Rules are used for a variety of purposes,

**1. Automatically Creating Activities:**

* When a new claim is created, an event fire rule can be used to automatically create an activity for an adjuster to review the claim.
* For example, "When a claim is created, fire an event rule to create a review activity."

**2. Sending Notifications or Emails:**

* When a claim status is updated to "settled," the system can trigger an event to send an email to the policyholder notifying them of the settlement.
* For example, "When a claim status changes to 'Approved,' send an email notification to the policyholder."

**3. Initiating Workflows:**

* When a claim loss cause changes, an event fire rule can initiate a workflow to investigate the change and determine if further action is needed.

**4. Updating Fields Automatically:**

* A rule can be triggered to automatically update certain fields based on an event. For example, when a claim is marked as "Closed," the system might automatically set the "Claim Closed Date" field to the current date.

**5. Logging or Auditing:**

* When a critical change happens (e.g., a policyholder’s name changes), an event rule could be fired to log this event for audit purposes, keeping track of all updates for compliance.

**Display Key:**

A display key represents a single user-viewable text string. Guidewire strongly recommends that any string literal that can potentially be seen by a user be defined as a display key rather than as a hard-coded String literal. Used is labels in PCF and where you want the string need to show there we can use it.

**Script parameter:**

Script parameters are resources defined in Guidewire Studio that you can use as global variables in Gosu code. To see the defined script parameters for your installation, navigate to the **Administration** tab on the **Utilities** > **Script Parameters** screen in Guidewire ClaimCenter. The detailed view of each script parameter displays a description of that script parameter. In the base configuration, file display.properties contains the display keys for script parameter descriptions in U.S. English.