**Guidewire important question and answer’s part 6:**

**1.Entity enhancement and Entity extensions:**

**Entity enhancement** is to add new properties and methods to Guidewire entities. Enhancements have the setters, getters and methods.

**Example:**

* Let's say we want to enhance the Claim entity to include a custom method that calculates the total payout amount (including claim amount and reserves).
* Create claimenhance.etx to include the entity.

**Getter:** A method that retrieves properties value. It will return the value.

**Setter:** A method that updates the properties value. It will set the value.

**Gosu Enhancement:**

Gosu enhancements provide additional methods (functionality) on a Guidewire entity. For example, suppose that you create an enhancement to the Activity entity. Within this enhancement, you add methods that support new functionality. Then, if you type Activity. (Activity followed by the dot character) within any Gosu code, Studio uses code completion on the Activity entity and displays any methods that you have defined in your activity enhancement, along with the native Activity entity methods.

Gosu enhancement files end in .gsx. Studio stores enhancement files in the gsrc folder hierarchy in the configuration folder.

**2.Load factor:**

Load factor is a powerful control for managing how much work is assigned to a particular user or group. It can be used at both the user and group level.

For example, if a user is on vacation or on leave, set their load factor to zero. Conversely, if the load factor for a group or user is a non-zero value, it enables proportional assignment to that group or user. You can also use load factor to adjust for part-time or reduced hours staff.

**3.XML and GX model:**

**XML model:**

XML Model refers to the XML-based representation of the data model, which defines the structure of entities, fields, relationships, and configurations. This XML is used for:

* Schema Definition (Database tables, columns, and relationships)
* UI Configuration (Screens, fields, and layouts)
* Integration (Message structures for web services, REST APIs, or file-based imports/exports)

**GX model:**

The GX (Gosu XML) Model is a hybrid of XML and Gosu logic used define dynamic UI behavior, conditional rules, and data transformations without full Gosu coding.

**Key property:**

A key property is typically used as a unique identifier for an object. For Guidewire business entities, the Public ID property is the property that identifies the entity to an external system. Accordingly, the Public ID property is commonly used as the key property for a GX model.

**Normal property:**

However, the normal property of a GX model does not have to uniquely identify an object. For example, a type might have an enhancement property that calculates a value using a complicated formula. If providing the final value to an external system is desired, the property can be mapped as a normal property.

**4.XSD editor:**

An XSD (XML Schema Definition) Editor is a tool used to define, edit, and validate XML schemas (.xsd files). In Guidewire Claim Center, XSD files are used for data modeling, defining web services, and managing integration mappings between Claim Center and external systems.

* A schema language that defines:
  + Rules for how XML documents must be structured.
  + Data types (e.g., string, date, integer).
  + Constraints (required fields, allowed values).
* Used in Claim Center for:
  + Validating XML messages (e.g., SOAP requests).
  + Ensuring data integrity in integrations.

**6. workload:**

Workload is a queue management system that automatically distributes new claims and tasks to adjusters based on:

* **Current capacity** (how many claims they're already handling)
* **Skills/Specializations** (auto, property, workers' comp)
* **Priority levels** (high-value claims vs. routine claims)

**7.Wizard:**

Wizards in Insurance Suite consist of a sidebar and workspace with users making selections and filling out information in the main workspace. Not all steps are mandatory, and users can return to a previous step to make changes as necessary.

**Use wizards for:**

* Tasks requiring a prescribed sequence such as filing a new claim.
* Complex tasks that users can complete more easily when the process is divided into a series of smaller and simpler steps.

**8. Foreign key (Many to One)**

* A **foreign key** is a field in one table (or entity) that **refers to the primary key** in **another table (entity)**.
* It is used to create a **relationship (link)** between two entities.

**Example:**

**Claim → Policy -** claim.Policy (Claim table has policy\_id column)

**Transaction → Check -** transaction.Check (Transaction table has check\_id column)

**Exposure → Incident -** exposure.Incident (Exposure table has incident\_id column)

**Array key (One to Many):**

* Auto-generated inverse of a foreign key (parent → children).
* No physical column – logical grouping.

**Example:**

**Policy → Claims** - policy.Claims (All claims linked to this policy)

**Check → Transactions** - check.Transactions (All transactions for this check)

**Incident → Exposures -** incident.Exposures (All exposures under this incident)

**Edge Foreign Key (Many to Many):**

* An edge foreign key is handling many-to-many relationships with additional data about the connection itself.
* Entity types A and B have foreign key references to each other. The foreign keys create a circular reference. Suppose that a bundle contains a new instance of A and a new instance of B. The circular reference would cause a foreign key constraint to fail upon committing the bundle. If ClaimCenter commits A before B is committed and in the database, a constraint failure occurs on the foreign key from A to B. The converse order of committing B before A causes a similar failure.
* An edge foreign key in place of a standard foreign key resolves circular references so ClaimCenter can determine a safe order for committing the entity instances within a cycle. An edge foreign key from A to B introduces a new, hidden associative entity with a foreign key to A and a foreign key to B. The edge foreign key associates A and B without establishing foreign keys in the database directly between them. With an edge foreign key, ClaimCenter can safely first commit new object A, then new object B, and finally the edge foreign key instance.

**Example:**

* **Transaction ↔ Claim** - TransactionClaimEdge table with transaction\_id, claim\_id, plus allocation\_amount
* **Check ↔ Payment** - CheckPaymentEdge table with check\_id, payment\_id, plus clearing\_date
* **Contact ↔ User**  - ContactUserEdge table with contact\_id, user\_id, plus relationship\_type

**10. Plugins:**

Claim Center plugins are classes that Claim Center invokes to perform an action or calculate a result at a specific time in its business logic. Claim Center defines plugin interfaces for various purposes.

**Uses:**

* Define how Claim Center sends messages to external systems.
* Perform calculations.
* Generate new data for other application logic, such as generating a new claim number.
* Define how the application interacts with other Guidewire Insurance Suite applications for important actions relating to claims, policies, billing, and contacts.
* Define how the application interacts with other third-party external systems such as a document management system, third-party claim systems, third-party policy systems, or third-party billing systems.
* Provide configuration points for your own business logic at clearly defined places in application operation, such as validation or assignment.

1. **Transport plugin:**

* It is an essential component that enables communication between guidewire applications and external systems.
* It is crucial for integrating Guidewire applications (like Policy Center, Claim Center, and Billing Center) with other external systems, such as third-party services, databases, or other enterprise applications.

1. **After send plugin:**

* The After Send Plugin is designed to execute custom logic after a message has been sent from a Guidewire application (such as Policy Center, Billing Center, or Claim Center) to an external system or service.
* It is often used to handle additional tasks or workflows based on the message’s sending outcome, such as logging, error handling, or updating application data.

1. **Request plugin:**

* The Request Plugin is responsible for sending a request from Guidewire to an external system. It defines the structure of the message, handles the preparation and transmission of the request, and ensures that the correct data is sent to the target system.

1. **Reply plugin:**

* The Reply Plugin handles the processing of the response received from the external system. Once the external system has processed the request and returned a reply, the Reply Plugin is invoked to parse the response, interpret the data, and trigger the appropriate actions within Guidewire based on the response.

**11.Messaging stages:**

1. **Application startup –** At application startup, ClaimCenter checks its configuration information and constructs messaging destinations. Each destination registers for specific events for which it wants notifications.
   1. In the hypothetical situation, the destination registers for the CheckAdded and CheckChanged events.
2. **Users and APIs trigger events –** Events trigger after data changes. For example, a change to data in the user interface, or an outside system calls a web service that changes data. The event represents the changes to application data as the entity commits to the database.
3. **Event Fired rule set is executed –** ClaimCenter runs the appropriate Event Fired rule set for each message destination defined for the event. triggers. A rule set can choose to generate new messages. Messages have a text-based message payload.
   1. For example, you might write rules that check if the event name is CheckAdded. When such an event is identified, the rules might generate a message with an XML payload that describes the added check.
4. **ClaimCenter sends message to a destination –** Messages are put in a queue and handed one-by-one to the messaging destination.
   1. In ClaimCenter, a check printing destination might take an XML payload and submit the message to an external message queue to notify the external system of the check.
5. **ClaimCenter waits for an acknowledgment –** The external system replies with an acknowledgment to the destination after it processes the message, and the destination’s messaging plugins process this information. If the message was successfully sent, the messaging plugins submit an ACK, and ClaimCenter sends the next message.

**12. GOSU templates:**

Gosu templates are reusable code patterns that simplify common development tasks in ClaimCenter. They combine Gosu (Guidewire's Java-like language) with XML-like placeholders for dynamic content.

Example:

1. PCF templates
2. Message templates
3. Document templates

**13. Policy administration system:**

In the Claim Center base configuration, you can do the following:

* Retrieve a policy snapshot in the process of gathering claim information – It is necessary for Claim Center to have a verified policy to ultimately be able pay on a claim. For example, in the New Claim wizard when starting a new claim, you must either find a policy or generate an unverified policy.
* View a policy in Policy Center – The integration opens an instance of Policy Center or any policy administration system that has a web interface.
* Claim Center sends large loss notifications to Policy Center – Sending notifications to the policy’s underwriter helps in determining risk and ultimately granting a policy renewal.

A diagram of a policy system

AI-generated content may be incorrect.