Salesforce System Integration Interview Prep

Remote Process Invocation – Request and Reply

Best solutions

1. **External Services** – invokes a REST API call – allows to invoke externally hosted service in declarative manner. External REST service are available in a an OpenAPI or Integrant schema definition. Contains primitive data types, nested objects not supported. Transaction invoked from Lightning Flow. Integration transaction doesn't risk exceeding the synchronous Apex governor limits.
2. **Salesforce Lighting –** consume WSDL and generate Apex proxy. Enable HTTP (REST) services, GET, PUT, POST, DELETE methods.
3. **Custom Visualforce page or button initiates Apex HTTP callout** – user initiated action calls and Apex action that executed this proxy Apex class

Suboptimal

1. **Trigger –** calls must be asynchronous (@future)
2. **Batch job callout –** bundle responses together and make 1 callout for every 200 records processed.

Endpoint capability

Endpoint should be able to receive a web services call via HTTP. Salesforce must be able to access endpoint via the internet, curl or postman. Apex SOAP callout- WSDL 1.1, SOAP 1.1

Apex HTTP callout -REST service using standard GET, POST, PUT, DELETE methods

Data Volumes

Small volume, real time activities, due to small timeout values and maximum size of the request or response from Apex call solution.

Timeliness

1. Request is typically invoked from user interface, must not keep user waiting
2. Governor limit of 120 second timeout for callouts
3. Remote process needs to be completed in SF limit and user expectations else seen as slow system
4. Apex synchronous apex limits are 10 transaction than run for 5 seconds

    - Make sure external server callout is less than 5 seconds

State management

1. Salesforce stores the external system External Id for specific record
2. The remote system stores the Salesforce unique record ID or other unique key

Security

Any call to external service should maintain:

1. Confidentiality
2. Integrity
3. Availability

Apex SOAP and HTTP callouts security considerations

1. One way SSL is enabled by default
2. 2 way SSL is enabled by self-signed certificate or CA-signed certificate
3. WS-Security is not supported
4. If necessary use one way hash or digital signature using Apex Crypto to ensure message integrity
5. Remote system must be protected by appropriate firewall mechanism

Error Handling – error occurs in form of error http error codes

400 – Bad request, 401 – Unauthorized, 404 – Not Found, 500 – Internal server error

Recovery – changes not committed to Salesforce until successful response, retry

x amount of times until success received else log error.

Idempotent Design consideration – duplicate calls when something goes wrong needs

need to have a message ID to make sure duplicates are not created

```
Http h = new Http();
HttpRequest req = new HttpRequest();
req.setEndpoint(url); //
req.setMethod('GET'); //

//Basic Authentication - specify in Named Credentials so no code change needed
String username = 'myname';
String password = 'mypwd';

Blob headerValue = Blob.valueOf(username + ':' + password);
String authorizationHeader = 'Basic ' +
EncodingUtil.base64Encode(headerValue);
req.setHeader('Authorization', authorizationHeader);

HttpResponse response = h.send(req);

if (response.getStatusCode() == 200) {
    Map&amp;amp;amp;lt;String, Object&amp;amp;amp;gt; results = (Map&amp;amp;amp;lt;String, Object&amp;amp;amp;gt;)
JSON.deserializeUntyped(response.getBody());
    List&amp;amp;amp;lt;Object&amp;amp;amp;gt; animals = (List&amp;amp;amp;lt;Object&amp;amp;amp;gt;) results.get('animals');
    System.debug('Received the following animals:');
    for (Object animal: animals) {
        System.debug(animal);
    }
}

//Implement httpCalloutMock
global class YourHttpCalloutMockImpl implements HttpCalloutMock {
    global HTTPResponse respond(HTTPRequest req) {


Test.setMock(HttpCalloutMock.class, new YourHttpCalloutMockImpl());
```

Remote Process Invocation – Fire and forget

Best solutions

1. **Process-driven platform events** – using process builder/workflow rules to create process to publish insert or update event
2. **Event messages** – the process of communicating changes and responding to them without writing complex logic. One or more subscribers can listen to the same event and carry out actions through CometD.
3. **Customized-driven platform events** – events are created by Apex triggers or batch classes
4. **Workflow driven outbound messaging** – remote process is invoked from an insert or update event. Salesforce provides workflow-driven outbound messaging capability that allow sending SOAP messages to remote systems.
5. **Outbound messaging and callbacks** – callback helps mitigate out-of-sequence messaging also 2 things Idempotency and Retrieve of more data. When creating a new record and update externalId with the recordId from external system.

Suboptimal

1. **Lightning components** – user interface based scenarios, must guarantee delivery of message in code
2. **Triggers** – Apex triggers to perform automation based on records changes
3. **Batch jobs** – calls to remote system can be performed from a batch job. Solution allows batch remote process execution and for processing of reposes from remote system.

```
MyEventObject__e newEvent = new MyEventObject__e();
newEvent.Name__c = 'Test Platform Event';
newEvent.Description__c = 'This is a test message';
List&lt;Database.SaveResult&gt; results = EventBus.publish(newEvent);


Test.startTest();
// Create test events
// ...
// Publish test events with EventBus.publish()
// ...
// Deliver test events
Test.getEventBus().deliver();
// Perform validation
// ...
Test.stopTest();


//Listener - EMPConnector use login to and create EmpConnector(bayeuxParameters)
import static com.salesforce.emp.connector.LoginHelper.login;
//create consumer
SalesforceEventPayload eventPayload = new SalesforceEventPayload();
    Consumer&lt;Map&lt;String, Object&gt;&gt; consumer = event -&gt; {
            eventPayload.setPayload(event);
            itsReqHandler.handleRequest(eventPayload.getPayload());
};

//add subscription to event
subscription = empConnector.subscribe("/event/" + EVENT_NAME, replayFrom, consumer).get(WAIT_TIME, TimeUnit.SECONDS);
```

Error handling and recovery

Error handling – because this pattern is asynchronous the callout of event publish handles need to be handles as well as the remote system needs to handle queuing, processing and error handling.

Recovery

More complex, need some retry strategy if no retry is receive in the QoS (Quality of Service) time period

Idempotent Design considerations

Platform events are published once, there is no retry on the Salesforce side. It is up to the ESB to request that the events be replayed. In a replay, the platform events reply ID remains the same and ESB can try to duplicate messages based on reply ID.

Unique Id for outbound messages is sent and needs to be tracked by the remote system to make sure duplicate messages or events are not repossessed.

Security

1. **Platform events** – conforms to the security of the existing Salesforce org
2. **Outbound messages** – One way SSL enabled, two way SSL can be used with outbound message certificate. Whitelist the IP ranges of the remote integration servers, remote server needs appropriate firewalls
3. **Apex callout** – one way SSL enabled, 2 way SSL through self signed or CA signed certificates. Use one way hash or digital signature (Apex Crypto class) to ensure integrity of message. Remote system protected by appropriate firewall mechanisms.

Timeliness

Less important, control handed back to client immediate or after successful deliver message. Outbound message acknowledgment must occur in 24 hours (can be extended to seven days) otherwise message expires.

Platform events – send events to event bus and doesn't wait for confirmation or acknowledgement from subscriber. If subscribe doesn't pick up the message they can reply the event using replayID. High volume message are stored for 72 hours (3 days). Subscriber use CometD to subscribe to channel.

State management

Unique record identifiers are important for ongoing state tracking:

1. Salesforce store remote system primary or unique surrogate key for remote record
2. Remote system store Salesforce unique record ID or some unique surrogate key

Governor Limits

Limits depends on the type of outbound call and timing of the call

Reliable Messaging

1. **Platform Events** – form of reliable messaging. Salesforce pushed the event to subscribers. If the message doesn't gets picked up it can be replayed using reply ID.
2. **Apex callout** – recommend that the remote system implement JMS, MQ however it doesn't guarantee delivery to remote system, specific techniques such as processing positive acknowledgement from remote endpoint in addition to custom retry logic must be implemented.
3. **Outbound messaging** – if no positive acknowledgment receive, retry up to 24 hours. Retry interval increase exponentially 15 sec to 60 min intervals.

Publisher and subscriber not in same transaction

Publish event before committed to the database, subscriber receives the event and does lookup to not find the record.

Publish behavior – publish immediately/publish after commit

Publish events – @future or @queueable to callout to events only when commit has complete

Message sequencing

Remote system discard message with duplicate message ID

Salesforce send RecordId, remote system makes callback to Salesforce

Handing Deletes

Salesforce workflow can't track deletion of records, can't call outbound message for deletion. Workaround:

1. Create custom object called Deleted_Records__c
2. Create trigger to store info (unique identifier) in custom object
3. Implement workflow rule to initiate message based on the creation of custom object

Batch Data Synchronization

Best solutions

1. **Salesforce change data capture (Salesforce master)**
2. **Replication via third party ETL Tool (Remote system master) –** Bulk API
3. **Replication via third party ETL Tool (Salesforce master) –** SOAP API getUpdated()

Suboptimal

1. **Remote call-in** – call into SF, causes lots of traffic, error handling, locking
2. **Remote process invocation** – call to remote system, causes traffic, error handing, locking

Extract and transform accounts, contacts, opportunities from current CRM to Salesforce (initial data load import)

Extract, transform, load billing data into Salesforce from remote system on weekly basis (ongoing)

Data master

Salesforce or Remote system

Salesforce Change Data Capture

Publish insert, update, delete, undelete events which represents changes to Salesforce. Receive near real time changes of records and sync to external data store.

Takes care of continuous synchronization part, needs integration app to receive events and perform update to external system

Channel – /data/{objectName}_Change

Error handling – Pattern is async remote system should handle message queuing, processing and error handling.

Recovery – initiate retry based on service quality of service requirement. Use replyID to reply stream of events. Use CometD to retrieve past messages up to 72 hours.

Bulk API – Replication via 3rd party ETL tool (more than 100 000 records)

Allow to run change data capture against source data. Tool reacts to change in the source data set, transforms the data and them call Salesforce Bulk API to issue DML statement, can also use SOAP API.

1. Read control table to determine last time job ran, other control values needed
2. Use above control values to filter query
3. Apply predefine processing rules, validation rules, enrichments
4. Use available connectors/transformation capability to create destination data set
5. Write the data to Salesforce objects
6. If processing is success update the control variable
7. If process fail update control variable for process to restart

Consider

1. Chain and sequence ETL jobs to provide cohesive process
2. Use primary key for both systems to match incoming data
3. Use specific API methods to extract only updated data
4. Importing master-detail or lookup, consider using the parent key at the source to avoid locking. Group contacts for an account to be imported at the same time.
5. Post-import processing should only process data selectively
6. Disable Apex triggers, workflow and validation rules
7. Use the defer calculations permission to defer sharing calculations until all data loaded

Error handling

If error occur during read operation, retry for errors. If errors repeat implement control tables/error tables in context of:

1. Log the error
2. Retry the read operation
3. Terminate if successful
4. Send a notification

Security

1. Lightning Platform license with at least "API Only" user permission
2. Standard encryption to keep password access secure
3. Use HTTPS protocol

Timeliness

Take care to design the interface so all batch processes complete in a designated batch window

Loading batches during business hours not recommended

Global operations should run all batch processes at the same time

State management

Use surrogate key between two systems.

Standard optimistic locking occurs on platform and any updates made using the API require the user who is editing the record to initiate a refresh and initiate their transaction. Optimistic locking means:

1. Salesforce doesn't maintain state of record being edited
2. Upon read, records time when data was extracted
3. If user updated the record and before save checks if another user has updated
4. System notified user update was made and use retrieve the latest version before updating

Middleware capabilities

Middleware tool that supports Bulk API

Supports the getUpdated() function – provide closest implementation to standard change data capture capability in Salesforce

Extracting data

Use the getUpdated() and getDeleted() SOAP API to sync an external system with Salesforce at intervals greater than 5 minutes. Use outbound messaging for more frequent syncing.

When querying can return more than an million results, consider the query capability of the Bulk API.

Remote Call-In

Best solutions

1. **SOAP API –** Publish events, query data, CRUD. Synchronous API, waits until it receives a response. Generated WSDL – enterprise (strongly-typed), partner (loosely typed). Must have a valid login and obtain session to perform calls. Allows partial success if the records are marked with errors, also allows "all or nothing" behavior.
2. **REST API –** Publish events, query data, CRUD. Synchronous API, waits until it receives a response. Lightweight and provides simple method for interacting with Salesforce. Must have a valid login and obtain session to perform calls. Allows partial success if the records are marked with errors, also allows "all or nothing" behavior. Output of one to use as input to next call.

Suboptimal

1. **Apex Web services –** use when: full transaction support is required, custom logic needs to be applied before commenting.
2. **Apex REST services –** lightweight implementation of REST services
3. **Bulk API –** submitting a number of batches to query, update, upsert or delete large number of records.

Authentication

Salesforce supports SSL (Security Socket Layer) and TLS (Transport Later Security), Ciphers must be at least length 128 bits

Remote system has to authenticate before accessing any REST service. Remote system can use OAuth 2.0 or username/password. Client has to set the authorization HTTP header with the appropriate value.

Recommend client caches the session ID rather than creating a new session ID for every call.

Accessibility

Salesforce provides a REST API that remote system can use to:

1. Query data in org
2. Publish events to org
3. CRUD of data
4. Metadata

Synchronous API

After call to server it waits for a response, asynchronous call to Salesforce is not supported

REST vs SOAP

REST exposes resources as URI and uses HTTP verbs to define CRUD operations. Unlike SOAP, REST required no predefined contract, utilize XML and JSON for responses, and has loosely typed. Advantage includes ease of integration and great use for mobile and web apps.

Security

Client executing the REST needs a valid Salesforce login and obtain a access token. API respects the object and field level security for the logged in user

Transaction/Commit Behavior

By default every record is treated as a separate transaction and committed separately. Failure of one records does not cause rollback of other changes. Using the composite API makes a series of updates in one call.

Rest composite resource

Perform multiple operation in a single API call. Also use output of one call to be input of next call. All response bodies and HTTP statuses are returned in a single response body. The entire requires counts as single call towards API limit.

Bulk API

For bulk operations use Rest-based BULK API

Event driven architecture

Platform events are defined the same way as you define a Salesforce object. Publishing to event bus is same as inserting Salesforce record. Only create and insert is supported.

Error handling

All remote call-in methods or custom API require remote system to handle any subsequent errors such as timeouts and retries. Middleware can be used to provide logic for recovery and error handling

Recovery

A custom retry mechanism needs to be created if QoS requirements dictate it. Important to consider impotent design characteristic.

Timeliness

Session timeout – session timeout when no activity based on SF org session timeout

Query timeout – each query has a timeout limit of 120 seconds

Data volumes

CRUD – 200 records per time

Blob size – 2GB ContentVersion (Chatter)

Query – query(), queryMore() return 500 records, max 2000

State management

Salesforce stores remote system primary key or unique key for the remote record

The remote system stores the Salesforce ID unique record ID or some unique

Governor limits

5000 API calls per 24 hour

10 query cursors open at time

Reliable messaging

Resolve the issue that delivery of a message to remote system where the individual components may be unreliable. SOAP API and REST API are synchronous and don't provide explicit support for any reliable messaging protocols.

Data visualization

Best solutions

1. **Salesforce Connect**

Suboptimal

1. **1. Request and reply** – Salesforce web services APIs (SOAP or REST) to make ad-hoc data requests to access and update external system data

Access data from external sources along with Salesforce data. Pull data from legacy systems, SAP, Microsoft, Oracle in real time without making a copy of the data.

Salesforce connect maps data tables in external systems to external objects in your org. External objects are similar to custom objects, except they map to data located outside SF org. Uses live connection to external data to keep external objects up to date.

Salesforce connects lets you

1. Query data in a external system
2. CRUD data in external system
3. Define relationships between external objects and standard or custom objects
4. Enable Chatter feed on external object page for collaboration
5. Run reports on external data

Salesforce Connect Adapters

OData adapter 2.0 or OData adapter 4.0 **–** connects data to exposed by any OData 2.0 or OData 4.0 producer

Cross-org adapter – connects to data that's stored in another Salesforce org. Used the Lightning Platform REST API

Custom adapter created via Apex – develop own adapter with the Apex Connector framework

Calling mechanism

External Objects – maps SF external objects to data tables in external systems. Connect access the data on demand and in real time. Provides seamless integration with Lightning Platform can do global search, lookup relationships, record feeds.

Also available to Apex, SOSL, SOQL queries, Salesforce API, Metadata API, change sets and packages.

Error handling

Run Salesforce Connector Validator tool to run some common queries and notice error types and failure causes

Benefits

1. Doesn't consume data storage in SF
2. Don't have to worry about regular sync between systems
3. Declarative setup can be setup quickly
4. Users can access external data with same functionality
5. Ability to do federated search
6. Ability to run reports

Considerations

Impact reports performance

Security considerations

Adhere to Salesforce org-level security, use HTTPS connect to any remote system.

OData understand behaviors, limitations and recommendations for CSRF (Cross-Site Request Forgery)

Timeliness

Request invoked by user interface, should not keep the user waiting

May take long to relieve data from external system, SF configured 120 sec maximum timeout

Completion of remote process should execute in timely manner

Data volumes

Use mainly for small volume, real time activities, due to small timeout and maximum size of request or response for Apex call solution.

State management

Salesforce stores primary or unique surrogate key for the remote record

Remote system store SF unique record ID or other unique surrogate key