# **Change Data Capture in Salesforce**

**What is change data capture?**

Change Data Capture is a mechanism that enables you to listen for changes to specific Salesforce records and react to those changes. When enabled for an object, it captures the changes made to the records, including the creation, modification, deletion, and undeletion events. Instead of repeatedly polling for changes, CDC pushes the data changes to subscribed applications in real-time using Salesforce Platform Events.

**Key Benefits of Change Data Capture:**

1. Real-time Data Sync: With CDC, you can instantly synchronize data across systems, reducing latency and ensuring that all applications have the latest information.
2. Event-Driven Architecture: CDC follows an event-driven approach, allowing applications to react to specific data changes rather than continuously querying for updates.
3. Efficient Resource Utilization: Traditional polling-based methods for data synchronization can be resource-intensive. CDC minimizes the number of API calls and increases the efficiency of your integration architecture.
4. Improved Data Accuracy: Real-time data updates eliminate the risk of stale data and ensure that everyone is working with the most current information.
5. Scalability: CDC is scalable and suitable for applications that require high throughput and low latency data propagation.

**Implementing Change Data Capture using Lightning Web Component (LWC)**

Step 1: Enable Change Data Capture for the Object

First, we need to enable Change Data Capture for the Salesforce object we want to track. Let's assume we want to track changes for the "Account" object.

* Go to Setup in your Salesforce org.
* In the Quick Find box, type Change Data Capture and select Change Data Capture from the results.
* Click on the Enable button for the "Account" object.

Step 2: Make a Lighting component

HTML:

<template>

<lightning-card title="EmpApi Example" icon-name="custom:custom14">

<div class="slds-m-around\_medium">

<p>

Use the buttons below to subscribe and unsubscribe to a

streaming channel!

</p>

<lightning-input

label="Channel Name"

value={channelName}

onchange={handleChannelName}

></lightning-input>

<lightning-button

variant="success"

label="Subscribe"

title="Subscribe"

onclick={handleSubscribe}

disabled={isSubscribeDisabled}

class="slds-m-left\_x-small"

></lightning-button>

<lightning-button

variant="destructive"

label="Unsubscribe"

title="Unsubscribe"

onclick={handleUnsubscribe}

disabled={isUnsubscribeDisabled}

class="slds-m-left\_x-small"

></lightning-button>

</div>

</lightning-card>

</template>

Javascript:

import { LightningElement } from 'lwc';

import {

subscribe,

unsubscribe,

onError,

setDebugFlag,

isEmpEnabled,

} from 'lightning/empApi';

export default class EmpApiLWC extends LightningElement {

channelName = '/data/AccountChangeEvent';

isSubscribeDisabled = false;

isUnsubscribeDisabled = !this.isSubscribeDisabled;

subscription = {};

// Tracks changes to channelName text field

handleChannelName(event) {

this.channelName = event.target.value;

}

// Initializes the component

connectedCallback() {

// Register error listener

this.registerErrorListener();

}

// Handles subscribe button click

handleSubscribe() {

// Callback invoked whenever a new event message is received

const messageCallback = function (response) {

console.log('New message received: ', JSON.stringify(response));

// Response contains the payload of the new message received

};

// Invoke subscribe method of empApi. Pass reference to messageCallback

subscribe(this.channelName, -1, messageCallback).then((response) => {

// Response contains the subscription information on subscribe call

console.log(

'Subscription request sent to: ',

JSON.stringify(response.channel)

);

this.subscription = response;

this.toggleSubscribeButton(true);

});

}

// Handles unsubscribe button click

handleUnsubscribe() {

this.toggleSubscribeButton(false);

// Invoke unsubscribe method of empApi

unsubscribe(this.subscription, (response) => {

console.log('unsubscribe() response: ', JSON.stringify(response));

// Response is true for successful unsubscribe

});

}

toggleSubscribeButton(enableSubscribe) {

this.isSubscribeDisabled = enableSubscribe;

this.isUnsubscribeDisabled = !enableSubscribe;

}

registerErrorListener() {

// Invoke onError empApi method

onError((error) => {

console.log('Received error from server: ', JSON.stringify(error));

// Error contains the server-side error

});

}

}

Step 3: Now place the component wherever you like.

Now, you can see the result in web console in JSON format. It would look like this

{ "schema": "-pszPCNGM45tUPU1ftkjxEA", "payload": { "LastModifiedDate": "2020-01-25T20:36:12.000Z", "OwnerId": "005RM000001vI54mYAE", "CreatedById": "005RM000001vI54mYAE", "ChangeEventHeader": { "commitNumber": 65842604581, "commitUser": "005RM000001vI54mYAE", "sequenceNumber": 1, "entityName": "sale\_item\_\_c", "changeType": "CREATE", "changedFields": [], "changeOrigin": "com/salesforce/api/soap/47.0;client=SfdcInternalAPI/", "transactionKey": "00051c2e-a75a-3f97-03fc-cdf4e16d9d3c", "commitTimestamp": 1569443783000, "recordIds": [ "a00RM0000114ICTYA2" ] }, "CreatedDate": "2020-01-25T20:36:12.000Z", "name": "IPhone", "description": "IPhone black with triple camera", "LastModifiedById": "005RM000001vI54mYAE", }, "event": { "replayId": 15053 } }

* VISHAL KUMAR