Screen Pop: Architecture & Implementation

Overview

Screen pop is the automatic display of a customer record when they call. Agent's desktop automatically looks up the caller by phone number and displays their Salesforce record.

Benefits:

- No manual lookup needed (saves 30+ seconds per call)
- Agent sees customer context immediately
- Fewer wrong accounts selected
- Better first-call resolution

How It Works (The Flow)

```
Customer calls (ANI: +15551234567)
    ↓
Architect Flow receives call
    ↓
Data Action: Look up contact by phone
    in Salesforce API
    ↓
Salesforce returns: Contact ID + Account info
    ↓
Flow sets Interaction Attributes:
  - contact_id = "003xx000003SG"
  - account_id = "001xx000002Edc"
    ↓
Flow transfers to Support Queue
```

```
    ↓
Agent receives call
    ↓
Agent's desktop extension
    reads Interaction Attributes
    ↓
Desktop makes API call to Salesforce:
  GET /sobjects/Contact/003xx000003SG
    ↓
Browser pops Salesforce record in new window
    ↓
Agent sees: Name, account, cases, history
    ↓
Agent handles call with context
```

Architecture Components

1. Call Entry Point (Architect Flow)

The flow that receives inbound calls:

```
START
  ↓
Play: "Thank you for calling..."
  ↓
Data Action: lookup-contact-by-phone
  Input: ${interaction.caller.phoneNumber}
  Output: ${contact.id}, ${account.id}
  ↓
Decision: Contact found?
  YES → Set attributes → Transfer
  NO → Collect account number → Transfer
  ↓
Transfer to Queue
  (attributes go with call)
  ↓
```

```
END
```

2. Data Action (API Call)

```
{
  "name": "lookup-contact-by-phone",
  "method": "POST",
  "url": "https://your-instance.salesforce.com/services/apexrest/contact-lookup",
  "inputContract": {
    "phoneNumber": {
      "type": "string",
      "required": true
    }
  },
  "outputContract": {
    "success": { "type": "boolean" },
    "contactId": { "type": "string" },
    "firstName": { "type": "string" },
    "lastName": { "type": "string" },
    "accountId": { "type": "string" },
    "accountName": { "type": "string" },
    "tier": { "type": "string" }
  }
}
```

3. Interaction Attributes

Data attached to the call that agent's desktop can access:

```
{
  "contact_id": "003xx000003SG",
  "contact_name": "John Doe",
  "account_id": "001xx000002Edc",
  "account_name": "Acme Corp",
  "customer_tier": "Premium",
  "last_contact": "2026-03-10T14:30:00Z"
}
```

4. Agent Desktop Extension

JavaScript in the Genesys Workspace (desktop app or web):

```
// Listen for incoming interaction
genesysClient.on('interaction.incoming', async (interaction) => {
  // Get attributes set by flow
  const contactId = interaction.attributes?.contact_id;

  if (contactId) {
    // Pop Salesforce record
    const recordUrl = `https://your-instance.salesforce.com/${contactId}`;
    window.open(recordUrl, 'salesforce');
  }
});
```

Step-by-Step Implementation

Step 1: Create Salesforce Apex Endpoint

```
// ContactLookupController.cls

@RestResource(urlMapping='/contact-lookup')
global class ContactLookupController {
  @HttpPost
  global static LookupResponse lookup(String phoneNumber) {
    LookupResponse response = new LookupResponse();

    try {
      // Search for contact by phone
      List<Contact> contacts = [
        SELECT Id, FirstName, LastName, Email,
               AccountId, Account.Name,
               Custom_Tier__c
        FROM Contact
```

```
        WHERE Phone = :phoneNumber
           OR MobilePhone = :phoneNumber
        LIMIT 1
      ];

      if (contacts.isEmpty()) {
        response.success = false;
        return response;
      }

      Contact contact = contacts[0];
      response.success = true;
      response.contactId = contact.Id;
      response.firstName = contact.FirstName;
      response.lastName = contact.LastName;
      response.accountId = contact.AccountId;
      response.accountName = contact.Account?.Name;
      response.tier = contact.Custom_Tier__c;

    } catch (Exception e) {
      response.success = false;
      response.error = e.getMessage();
    }

    return response;
  }

  global class LookupResponse {
    public Boolean success;
    public String contactId;
    public String firstName;
    public String lastName;
    public String accountId;
    public String accountName;
    public String tier;
    public String error;
  }
}
```

Step 2: Create Architect Flow

In **Genesys Architect** → Create **Inbound Call Flow**:

Flow Steps:

```
1. START
   ↓
2. Play Audio: "Thank you for calling..."
   ↓
3. Data Action: lookup-contact-by-phone
   Input: ${interaction.caller.phoneNumber}
   ↓
4. Decision: ${dataAction.result.success}?

   IF YES:
   └─ Set Agent Variables:
      - contact_id = ${dataAction.result.contactId}
      - contact_name = ${dataAction.result.firstName} ${dataAction.result.lastName}
      - account_id = ${dataAction.result.accountId}
      - account_name = ${dataAction.result.accountName}
      - customer_tier = ${dataAction.result.tier}

   IF NO:
   └─ Play Audio: "Please hold while we locate your account..."

   ↓
5. Transfer to Queue: Support Queue
   ↓
6. DISCONNECT
```

Step 3: Create Desktop Extension

In **Genesys Cloud** → **Integrations** → **Custom Apps** → **Desktop App Extensions**:

```
// manifest.json
{
  "version": "1.0",
  "name": "Salesforce Screen Pop",
```

```
  "description": "Pops Salesforce contact on inbound call"
}

// index.html
<!DOCTYPE html>
<html>
<head>
  <script src="https://sdk.mypurecloud.com/v131/platform.min.js"></script>
</head>
<body>
  <script>
    const client = require('purecloud-platform-client-v2');

    // Initialize
    const platformClient = client.ApiClient.instance;
    platformClient.setEnvironment('mypurecloud.com');

    // Listen for incoming interaction
    const notificationService = platformClient.createNotificationService();

    notificationService.subscribe(
      'v2.conversations.{id}',
      async (event) => {
        const interaction = event.eventBody;

        // Check if attributes set by flow
        if (interaction.attributes?.contact_id) {
          const contactId = interaction.attributes.contact_id;
          const accountId = interaction.attributes.account_id;
          const contactName = interaction.attributes.contact_name;

          // Build Salesforce URL
          const baseUrl = 'https://your-instance.salesforce.com';
          const recordUrl = `${baseUrl}/${contactId}`;

          // Pop window
          window.open(recordUrl, 'salesforce_record',
            'width=1200,height=800,resizable=yes');

          console.log(`Screen pop: ${contactName} (${accountId})`);
```

```
      }
    }
  );
 </script>
</body>
</html>
```

Handling Edge Cases

Multiple Matches

Problem: Phone number found 3 contacts (different names, same company)

Solution: Let agent pick

```
Flow: Decision - ${dataAction.result.matches.length} > 1?

 IF YES:
 └─ Play: "Multiple accounts found. Press..."
 └─ Collect Input:
    "Press 1 for John Doe"
    "Press 2 for Jane Doe"
    "Press 3 for John Smith"
 └─ Set contact_id = ${dataAction.result.matches[input]}.id

 IF NO:
 └─ Continue normally
```

Contact Not Found

Problem: Phone number not in Salesforce

Solution: Offer fallback

```
Flow: Decision - ${dataAction.result.success}?
```

```
IF NO:
└─ Play: "Account not found. Please enter your account number..."
└─ Collect Input: Account Number
└─ Data Action: lookup-by-account-number
└─ Set attributes if found
└─ Transfer without pop if not found
```

Slow Lookup (Timeout)

Problem: Salesforce API slow, lookup takes 5+ seconds

Solution: Don't block the call

```
Flow: Data Action: lookup-contact-by-phone
  Timeout: 3 seconds

  Decision: Action succeeded?

    IF YES (within 3 sec):
    └─ Set attributes → Transfer

    IF NO (timed out):
    └─ Play: "Connecting you now..."
    └─ Transfer WITHOUT attributes
    └─ (Agent can manual search)
```

Salesforce Field Mapping

What agent sees after screen pop:

Data Point	Source	Salesforce Record
Contact Name	Flow attribute	Contact.Name
Email	API response	Contact.Email
Phone	API response	Contact.Phone
Account	Flow attribute	Account.Name
Tier/Priority	Flow attribute	Contact.Custom_Tier__c

Data Point	Source	Salesforce Record
Recent Cases	(automatic in SF)	Related Cases
Contact History	(automatic in SF)	Activity Timeline

Performance Optimization

Problem: Lookup taking 2+ seconds

Causes:

- Salesforce API slow
- Network latency
- SOQL query inefficient

Solutions:

1. **Add Index in Salesforce**

```
-- Make phone lookups fast
CREATE INDEX idx_contact_phone
ON Contact(Phone, MobilePhone);
```

2. **Cache recent lookups**

```
class ScreenPopCache {
  constructor(ttlSeconds = 3600) {
    this.cache = new Map();
    this.ttl = ttlSeconds;
  }

  async lookup(phone) {
    const cached = this.cache.get(phone);
    if (cached && Date.now() - cached.timestamp < this.ttl * 1000) {
      return cached.data;
    }

    const result = await callSalesforceAPI(phone);
    this.cache.set(phone, { data: result, timestamp: Date.now() });
```

```
    return result;
  }
}
```

3. **Use webhook instead of polling**
   - Salesforce creates contact → webhook updates Genesys
   - (More advanced, requires webhook setup)

Testing Screen Pop

Manual Test

1. Configure test flow in Architect
2. Set up desktop extension locally
3. Make test call to your Genesys DID
4. Verify:
   - ✓ Architect flow receives call
   - ✓ Data Action executes (check execution history)
   - ✓ Salesforce API returns contact
   - ✓ Flow sets interaction attributes
   - ✓ Desktop receives attributes
   - ✓ Window pops Salesforce record

Automated Test

```
// test-screen-pop.js

async function testScreenPop() {
  // 1. Test Salesforce lookup
  const contact = await lookupContactByPhone('+15551234567');
  console.assert(contact.id, 'Contact not found');

  // 2. Test Genesys Data Action
  const result = await callDataAction('lookup-contact-by-phone', {
    phoneNumber: '+15551234567'
  });
  console.assert(result.success, 'Data Action failed');
```

```
  // 3. Test flow
  const call = await makeTestCall('+15551234567');
  const attributes = call.attributes;
  console.assert(attributes.contact_id, 'No contact_id in attributes');

  console.log('✓ Screen pop test passed');
}
```

Troubleshooting

Problem: Desktop doesn't pop window

Check:

- ☐ Extension enabled in Genesys Workspace?
- ☐ Flow sets interaction attributes?
- ☐ Desktop JavaScript can access attributes?
- ☐ Browser allows popups?
- ☐ Salesforce URL correct?

Debug:

```
// Add logging to desktop extension
genesysClient.on('interaction.incoming', (interaction) => {
  console.log('Attributes:', interaction.attributes);
  console.log('Contact ID:', interaction.attributes?.contact_id);
});
```

Problem: Lookup returns "Contact not found" for valid phone

Check:

- ☐ Phone in Salesforce exactly matches flow phone?

- [ ] Phone format consistent (E.164, local, international)?
- [ ] Apex query correct?

Debug:

```
// Test in Salesforce Developer Console
List<Contact> contacts = [
  SELECT Id FROM Contact
  WHERE Phone = '+15551234567'
];
System.debug(contacts.size() + ' contacts found');
```

Problem: Performance is slow

Check:

- [ ] Data Action timeout too long?
- [ ] Salesforce API slow?
- [ ] Network latency high?

Optimize:

- Set timeout to 3 seconds (not 10)
- Add Salesforce index on Phone field
- Implement caching
- Use webhook instead of API call

Production Deployment Checklist

- [ ] Salesforce Apex endpoint created & tested
- [ ] Data Action configured in Genesys
- [ ] Architect flow configured & tested
- [ ] Desktop extension installed & enabled
- [ ] Error handling for timeouts
- [ ] Fallback for not-found contacts
- [ ] Field mapping verified

- [ ] Performance acceptable (<3 sec)
- [ ] Monitoring & alerting set up
- [ ] Staff trained on screen pop behavior

Related Topics

- Chapter 11: API Endpoints Reference (Salesforce API)
- Chapter 11: Error Handling & Retry Strategy
- Chapter 5: Data Actions (in Architect flows)
- Chapter 12: Contact Sync from Salesforce

Revision #1
Created 15 March 2026 00:44:40 by Cesar Gzz
Updated 15 March 2026 00:44:52 by Cesar Gzz