Activity Logging & Webhooks

Overview

Activity logging means capturing what happened during a call and writing it back to your CRM. After the call ends, Genesys sends details to Salesforce so agents can track customer interactions.

Benefit: Single source of truth. All customer interactions visible in CRM.

The Flow

```
Call Happens
    ↓
Agent handles call (duration, queue, notes)
    ↓
Call ends
    ↓
Genesys webhook: conversation.ended
    ↓
Your backend receives webhook
    ↓
Fetch conversation details:
  - Caller phone
  - Duration
  - Agent name
  - Recording URL
  - Start/end time
    ↓
Find matching Salesforce contact (by phone)
```

```
    ↓
Create Salesforce Task:
  Subject: "Call with John Doe"
  Description: "Duration: 10 min..."
  Recording: [link]
  WhoId: Contact ID
  WhatId: Account ID
    ↓
Update Contact:
  LastActivityDate = today
  Last_Call_Date = today
    ↓
Success! Call logged in CRM
```

Setup: Genesys Webhook

In Genesys Admin

1. Go to **Integrations → Webhooks**
2. Click **Add Webhook**
3. Configure:
   - **Event:** `conversation.ended`
   - **URL:** `https://your-server.com/webhook/call-ended`
   - **Payload:** Send call details
   - **Retry:** 3 times if fails

Webhook Payload (What Genesys Sends)

```
{
  "conversationId": "conversation-111",
  "participantId": "user-agent-1",
  "conversationType": "phone",
  "callerId": "+15551234567",
  "calleeId": "+18001234567",
  "direction": "INBOUND",
  "startTime": "2026-03-14T10:00:00Z",
```

```
  "endTime": "2026-03-14T10:10:30Z",
  "durationSeconds": 630,
  "recordingId": "recording-222",
  "agentId": "user-agent-1",
  "agentName": "Agent Smith",
  "queueId": "queue-456",
  "queueName": "Support Queue",
  "interactionId": "interaction-123",
  "attributes": {
    "contact_id": "003xx000003SG",
    "account_id": "001xx000002Edc",
    "customer_tier": "Premium"
  }
}
```

Implementation: Webhook Handler

Node.js Server

```
const express = require('express');
const axios = require('axios');
require('dotenv').config();

const app = express();
app.use(express.json());

/**
 * Webhook endpoint - Genesys calls this when call ends
 */
app.post('/webhook/call-ended', async (req, res) => {
  const {
    conversationId,
    callerId,
    durationSeconds,
    recordingId,
    agentName,
```

```
    queueName,
    startTime,
    endTime,
    attributes
  } = req.body;

  console.log(`□□ Call ended: ${conversationId}, Duration: ${durationSeconds}s`);

  try {
    // 1. Find matching Salesforce contact
    const contact = await findSalesforceContactByPhone(callerId);

    if (!contact) {
      console.warn(`⚠ Contact not found for ${callerId}`);
      return res.status(200).json({ message: 'Contact not found' });
    }

    // 2. Get recording URL
    let recordingUrl = null;
    if (recordingId) {
      recordingUrl = await getRecordingUrl(recordingId);
    }

    // 3. Create Task in Salesforce
    const task = {
      Subject: `Call with ${contact.Name}`,
      Description: `
Inbound Call from ${callerId}
Duration: ${Math.floor(durationSeconds / 60)} minutes
Agent: ${agentName}
Queue: ${queueName}
Start: ${startTime}
End: ${endTime}
${recordingUrl ? `Recording: ${recordingUrl}` : ''}
      `.trim(),
      WhoId: contact.Id,          // Contact ID
      WhatId: contact.AccountId,  // Account ID
      ActivityDate: new Date().toISOString().split('T')[0],
      CallDurationInSeconds: durationSeconds,
      CallType: 'Inbound',
```

```
      Status: 'Completed',
      Type: 'Call'
    };

    const taskResult = await createSalesforceTask(task);
    console.log(`✅ Task created: ${taskResult.id}`);

    // 4. Update Contact's LastActivityDate
    await updateSalesforceContact(contact.Id, {
      LastActivityDate: new Date().toISOString().split('T')[0],
      Last_Call_Date__c: new Date().toISOString(),
      Last_Call_Duration__c: durationSeconds
    });

    res.status(200).json({ taskId: taskResult.id });

  } catch (error) {
    console.error('❌ Failed to log activity:', error.message);
    res.status(500).json({ error: error.message });
  }
});

/**
 * Find Salesforce contact by phone number
 */
async function findSalesforceContactByPhone(phoneNumber) {
  const query = `
    SELECT Id, Name, AccountId, Email
    FROM Contact
    WHERE Phone = '${phoneNumber}'
       OR MobilePhone = '${phoneNumber}'
    LIMIT 1
  `;

  const response = await axios.get(
    `${process.env.SALESFORCE_INSTANCE}/services/data/v57.0/query`,
    {
      params: { q: query },
      headers: {
        'Authorization': `Bearer ${process.env.SALESFORCE_TOKEN}`,
```

```
        'Content-Type': 'application/json'
      }
    }
  );

  return response.data.records[0] || null;
}

/**
 * Get Genesys recording URL
 */
async function getRecordingUrl(recordingId) {
  const response = await axios.get(
    `https://api.mypurecloud.com/api/v2/recordings/${recordingId}/media`,
    {
      headers: {
        'Authorization': `Bearer ${process.env.GENESYS_TOKEN}`
      },
      maxRedirects: 0,
      validateStatus: status => status === 307  // Expect redirect
    }
  );

  // Returns presigned S3 URL in Location header
  return response.headers.location || null;
}

/**
 * Create Salesforce Task
 */
async function createSalesforceTask(taskData) {
  const response = await axios.post(
    `${process.env.SALESFORCE_INSTANCE}/services/data/v57.0/sobjects/Task`,
    taskData,
    {
      headers: {
        'Authorization': `Bearer ${process.env.SALESFORCE_TOKEN}`,
        'Content-Type': 'application/json'
      }
    }
```

```
  );

  return response.data;
}

/**
 * Update Salesforce Contact
 */
async function updateSalesforceContact(contactId, updateData) {
  const response = await axios.patch(
    `${process.env.SALESFORCE_INSTANCE}/services/data/v57.0/sobjects/Contact/${contactId}`,
    updateData,
    {
      headers: {
        'Authorization': `Bearer ${process.env.SALESFORCE_TOKEN}`,
        'Content-Type': 'application/json'
      }
    }
  );

  return response.data;
}

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Webhook server running on port ${PORT}`);
});
```

Data Mapping

Call → Salesforce Task

Genesys Data	Salesforce Field	Mapping
`callerId` (phone)	Task subject	"Call with {contact.Name}"
`durationSeconds`	`CallDurationInSeconds`	Direct
`startTime`	Task description	Include timestamp

Genesys Data	Salesforce Field	Mapping
`endTime`	Task description	Include timestamp
`recordingId`	Task description	Link to recording
`agentName`	Task description	"Agent: {name}"
`queueName`	Task description	"Queue: {name}"
Contact ID (from lookup)	`WhoId`	Links to contact
Account ID (from lookup)	`WhatId`	Links to account
Current date	`ActivityDate`	Today's date

Advanced: Logging with Custom Fields

Salesforce custom fields capture more data:

```
// Salesforce custom fields setup
const task = {
  // Standard fields
  Subject: `Call with ${contact.Name}`,
  WhoId: contact.Id,
  WhatId: contact.AccountId,

  // Custom fields
  Call_Type__c: 'Inbound',
  Call_Queue__c: queueName,
  Call_Agent__c: agentName,
  Call_Duration_Seconds__c: durationSeconds,
  Call_Recording_URL__c: recordingUrl,
  Call_Channel__c: 'Phone',
  Customer_Tier__c: attributes?.customer_tier,
  Was_Transferred__c: false,
  Call_Outcome__c: 'Completed',
  Agent_Notes__c: attributes?.notes
};
```

```
// Custom field names end with __c (Salesforce convention)
```

Error Handling

Problem: Contact Not Found

```
// Option 1: Skip (don't log if no contact)
if (!contact) {
  console.warn(`No contact for ${callerId}`);
  return res.status(200).json({ message: 'Contact not found' });
}

// Option 2: Create contact
if (!contact) {
  contact = await createSalesforceContact({
    LastName: callerId,  // Use phone as fallback
    Phone: callerId
  });
}

// Option 3: Log to generic "unknown caller" account
if (!contact) {
  contact = { Id: UNKNOWN_CALLER_ACCOUNT };
}
```

Problem: Recording URL Timeout

```
async function getRecordingUrlSafe(recordingId, timeoutMs = 3000) {
  try {
    const response = await axios.get(
      `https://api.mypurecloud.com/api/v2/recordings/${recordingId}/media`,
      {
        headers: { 'Authorization': `Bearer ${token}` },
        timeout: timeoutMs
      }
```

```
    );
    return response.headers.location || null;
  } catch (error) {
    if (error.code === 'ECONNABORTED') {
      console.warn('Recording URL timeout');
      return null;  // Don't block task creation
    }
    throw error;
  }
}
```

Problem: Task Creation Fails

```
async function createTaskWithRetry(task, maxAttempts = 3) {
  for (let attempt = 1; attempt <= maxAttempts; attempt++) {
    try {
      const result = await createSalesforceTask(task);
      console.log(`  Task created: ${result.id}`);
      return result;
    } catch (error) {
      if (attempt === maxAttempts) {
        console.error(`  Failed after ${maxAttempts} attempts`);
        throw error;
      }

      const delayMs = 1000 * Math.pow(2, attempt - 1); // Exponential backoff
      console.warn(`Retry in ${delayMs}ms...`);
      await sleep(delayMs);
    }
  }
}
```

Webhook Security

Verify Genesys Signature

Genesys signs webhooks so you can verify they're legitimate:

```
const crypto = require('crypto');

/**
 * Verify Genesys webhook signature
 */
function verifyWebhookSignature(request, secret) {
  const signature = request.headers['x-genesys-webhook-signature'];
  const timestamp = request.headers['x-genesys-webhook-timestamp'];

  if (!signature || !timestamp) {
    throw new Error('Missing signature or timestamp header');
  }

  // Reconstruct the signed string
  const body = request.rawBody; // Must be raw, not parsed
  const signedString = `${timestamp}.${body}`;

  // Calculate HMAC
  const hash = crypto
    .createHmac('sha256', secret)
    .update(signedString)
    .digest('base64');

  // Verify
  if (hash !== signature) {
    throw new Error('Webhook signature invalid');
  }

  return true;
}

// Middleware to verify all webhooks
app.use((req, res, next) => {
  req.rawBody = req.body;  // Keep raw body
  next();
});

app.post('/webhook/call-ended', (req, res, next) => {
```

```
  try {
    verifyWebhookSignature(req, process.env.GENESYS_WEBHOOK_SECRET);
    next();
  } catch (error) {
    console.error('❌ Webhook signature verification failed:', error.message);
    return res.status(401).json({ error: 'Unauthorized' });
  }
});
```

Monitoring & Alerts

```
class ActivityLoggingMonitor {
  constructor() {
    this.stats = {
      totalCalls: 0,
      logsCreated: 0,
      logsFailed: 0,
      contactsNotFound: 0,
      avgProcessingTime: 0
    };
  }

  recordSuccess(processingTimeMs) {
    this.stats.totalCalls++;
    this.stats.logsCreated++;
    this.updateAvgTime(processingTimeMs);
  }

  recordFailure(reason) {
    this.stats.totalCalls++;
    this.stats.logsFailed++;

    if (reason === 'contact_not_found') {
      this.stats.contactsNotFound++;
    }

    // Alert if too many failures
```

```
    const failureRate = this.stats.logsFailed / this.stats.totalCalls;
    if (failureRate > 0.05) { // > 5%
      this.alertHighFailureRate(failureRate);
    }
  }

  updateAvgTime(newTime) {
    const count = this.stats.logsCreated;
    this.stats.avgProcessingTime =
      (this.stats.avgProcessingTime * (count - 1) + newTime) / count;
  }

  reportHealth() {
    return `
Activity Logging Health:
  Total calls: ${this.stats.totalCalls}
  Logged: ${this.stats.logsCreated}
  Failed: ${this.stats.logsFailed}
  Not found: ${this.stats.contactsNotFound}
  Avg time: ${this.stats.avgProcessingTime.toFixed(0)}ms
  Success rate: ${((this.stats.logsCreated / this.stats.totalCalls) * 100).toFixed(1)}%
    `;
  }
}
```

Production Checklist

- ☐ Genesys webhook configured for `conversation.ended`
- ☐ Webhook URL is publicly accessible & secured
- ☐ Signature verification implemented
- ☐ Error handling for missing contacts
- ☐ Recording URL generation working
- ☐ Salesforce Task fields mapped
- ☐ Contact lookup by phone working
- ☐ Task creation tested with real calls
- ☐ Monitoring & alerting set up

- [ ] Staff trained (calls logged to CRM)
- [ ] Documentation updated

Related Topics

- Chapter 12: Screen Pop (related feature)
- Chapter 12: Contact Sync Patterns (keeping contacts in sync)
- Chapter 11: API Endpoints Reference (Genesys APIs)

Revision #1
Created 15 March 2026 00:45:25 by Cesar Gzz
Updated 15 March 2026 00:45:34 by Cesar Gzz