Module 1: Legacy System Analysis and Requirements Gathering

Executive Summary

This report analyzes the current VXML-based IVR system and details the requirements for its modernization through integration with Azure Communication Services ACS) and the Beckn Adaptive Protocol BAP) platform. The objective is to enable a seamless transition to modern technologies while maintaining continuous business operations.

1. Current System Architecture

System Overview

Our legacy IVR system is based on VXML 2.1 and consists of:

- **Voice Gateway:** Manages SIP trunking and call routing.
- **VXML Interpreter:** Processes voice flows using SRGS grammar for speech recognition.
- **Audio Prompts:** Relies on pre-recorded WAV files.
- **Backend Integration:** Connects to REST APIs for data.

Sample VXML Call Flow

The main menu prompt guides users:

- Press 1 for account balance.
- Press 2 for support.
- Press 3 for operator.

Prompt navigation is implemented via decision-based VXML grammar rules, with DTMF and basic speech recognition.

System Capabilities

- DTMF and limited vocabulary speech input
- Call transfer and conferencing
- Session data persistence
- Error handling and retry logic

System Limitations

- No real-time analytics
- No advanced Natural Language Understanding NLU
- Static call flows requiring manual updates
- Lack of multi-channel support

2. Integration Requirements

Azure Communication Services ACS Integration

Key capabilities required:

- Modern API-based call handling
- Advanced speech recognition through Azure Speech
- Call recording capabilities
- Push call analytics to Azure Monitor

Python Example ACS Client Setup):

```
from azure.communication.callautomation import CallAutomationClient
from azure.identity import DefaultAzureCredential

def setup_acs_client(endpoint):
    credential = DefaultAzureCredential()
    client = CallAutomationClient(endpoint=endpoint, credential=credential)
    return client
```

Beckn Adaptive Protocol BAP Integration

Key requirements:

- Service discovery and dynamic call routing
- Build BAP request context for each call
- Intent mapping from IVR inputs to BAP queries
- Track call fulfillment via BAP status

Node.js Example BAP Search Request):

```
const axios = require("axios");

async function searchServices(userIntent, callContext) {
    // ...set up BAP request...
    try {
        const response = await axios.post("https://gateway.beckn.network/search", bapRequ
        return response.data;
    } catch (error) {
        console.error("BAP search failed", error);
        return null;
    }
}
```

3. Technical Challenges and Constraints

Identified Key Challenges:

Challenge	Impact	Mitigation Strategy
VXML to Modern API Migration	High	Phased migration, parallel operation
Session State Management	Medium	Use Redis for distributed state
Speech Recognition Accuracy	High	Custom models on Azure Speech
Real-time BAP Integration	Medium	Add caching for frequent queries
Legacy Hardware Compatibility	Low	Abstract hardware layer via SIP gateway

Implementation Constraints:

- Minimum 99.9% uptime requirement during migration
- Max 500ms call routing response time
- Support for 1000+ concurrent calls
- Data retention: 90 days for logs and recordings
- PCI DSS, GDPR compliance (especially for payment IVR)s