ServiceNow A2A Testing - ServiceNow_A2A_Testing_Guide

ServiceNow A2A Testing Tools

September 2025

1 ServiceNow Agent-to-Agent (A2A) Testing Guide

Version: 1.0
Date: September 2025
Audience: ServiceNow Customers, Developers, and System Administrators

---

1.1 Table of Contents

1.2 Overview

This guide provides comprehensive tools and instructions for testing ServiceNow Agent-to-Agent (A2A) implementations. It includes both Python-based clients and curl-based testing scripts that work with any ServiceNow instance and agent.

1.2.1 What's Included

- **Generic Python A2A Client** - Production-ready client for any ServiceNow instance

- **Automated Testing Scripts** - Bash scripts for quick testing with curl
- **Configuration Templates** - Easy setup for different environments
- **Troubleshooting Guide** - Common issues and solutions

1.2.2 Key Features

✅ **Universal Compatibility** - Works with any ServiceNow instance and agent
✅ **Multiple Testing Methods** - Python SDK, direct HTTP, and curl
✅ **Comprehensive Error Handling** - Detailed error messages and debugging
✅ **Production Ready** - Suitable for development, testing, and production use

---

1.3 Prerequisites

1.3.1 System Requirements

- **Operating System:** macOS, Linux, or Windows with WSL
- **Python:** 3.8 or higher (for Python clients)
- **curl:** Available on most systems
- **jq:** Optional, for pretty JSON formatting

1.3.2 ServiceNow Requirements

- **ServiceNow Instance** with A2A capabilities
- **Agent ID** of the agent you want to test
- **API Key** with appropriate permissions for A2A operations

1.3.3 Python Dependencies

```
pip install httpx a2a-sdk
```

1.4 Quick Start

1.4.1 1. Download and Setup

```
# Create project directory
mkdir servicenow-a2a-testing
```

```
cd servicenow-a2a-testing

# Copy all provided files to this directory
# (generic_a2a_client.py, test_servicenow_agent.sh, etc.)
```

1.4.2 2. Configure Your Environment

```
# Copy the template
cp .env.template .env

# Edit .env with your ServiceNow details
SERVICENOW_INSTANCE=your-instance.service-now.com
SERVICENOW_AGENT_ID=your-agent-id
SERVICENOW_TOKEN=your-api-key
```

1.4.3 3. Run Your First Test

```
# Test with the automated script
./test_servicenow_agent.sh

# Or test with Python
python generic_a2a_client.py
```

1.5 Testing Tools

This guide provides three main testing approaches:

1.5.1 1. Automated Testing Script (`test_servicenow_agent.sh`)

- **Best for:** Quick validation and debugging
- **Features:** Full workflow testing, error handling, response parsing
- **Usage:** `./test_servicenow_agent.sh [options]`

1.5.2 2. Python Generic Client (`generic_a2a_client.py`)

- **Best for:** Integration into applications
- **Features:** Production-ready, configurable, conversation context
- **Usage:** Import as module or run directly

1.5.3 3. Curl Command Generator (`generate_curl_commands.sh`)

- **Best for:** Manual testing and debugging
- **Features:** Generates ready-to-use curl commands
- **Usage:** `./generate_curl_commands.sh [options]`

1.6 Configuration Guide

1.6.1 Environment Variables

Create a `.env` file in your project directory:

```
# ServiceNow Instance (without https://)
SERVICENOW_INSTANCE=your-company.service-now.com

# Agent ID (found in ServiceNow agent configuration)
SERVICENOW_AGENT_ID=your-agent-id-here

# API Key (generated in ServiceNow)
SERVICENOW_TOKEN=your-api-key-here
```

1.6.2 Finding Your Agent ID

1. Log into your ServiceNow instance
2. Navigate to **Agent Intelligence** > **Agents**
3. Select your agent
4. Copy the ID from the URL or agent details page

1.6.3 Generating an API Key

1. In ServiceNow, go to **System Web Services** > **REST API Explorer**
2. Or use the ServiceNow Developer Portal
3. Generate a new API key
4. Ensure it has permissions for A2A operations

1.7 Testing Methods

1.7.1 Method 1: Automated Testing Script

The automated script provides the most comprehensive testing:

```
# Basic test (uses .env configuration)
./test_servicenow_agent.sh

# Test with custom message
./test_servicenow_agent.sh -m "What services do you provide?"

# Verbose output for debugging
./test_servicenow_agent.sh -v

# Test only agent discovery
./test_servicenow_agent.sh --discovery-only

# Custom configuration
./test_servicenow_agent.sh -i company.service-now.com -a agent-id -t api-key
```

Sample Output:

```
[INFO] ServiceNow A2A Agent Test
==================================
Instance: company.service-now.com
Agent ID: your-agent-id
API Key:  your-api-key...
Message:  What can you help me with?
==================================

[INFO] Step 1: Testing agent discovery...
[SUCCESS] Agent discovery successful!
  Agent Name: Your Agent Name
  Description: Agent description here

[INFO] Step 2: Sending test message to agent...
[SUCCESS] Message sent successfully!

[INFO] Agent Response:
Response 1: Hello! I can help you with...

[SUCCESS] All tests completed successfully!
```

1.7.2 Method 2: Python Client

Use the Python client for programmatic access:

```
import asyncio
from generic_a2a_client import ServiceNowA2AClient
```

```
async def test_agent():
    # Option 1: Use environment variables
    client = ServiceNowA2AClient()

    # Option 2: Direct configuration
    # client = ServiceNowA2AClient(
    #     instance="your-instance.service-now.com",
    #     agent_id="your-agent-id",
    #     api_key="your-api-key"
    # )

    # Send a simple message
    response = await client.connect_and_send("What can you help me with?")
    print(response)

# Run the test
asyncio.run(test_agent())
```

Advanced Usage with Context:

```
async def conversation_test():
    client = ServiceNowA2AClient()

    # First message
    response1 = await client.connect_and_send("What services do you provide?")

    # Extract context for follow-up
    context_id = response1.get('result', {}).get('contextId')

    # Follow-up message with context
    response2 = await client.connect_and_send(
        "Tell me more about the first one",
        context_id=context_id
    )

    return response1, response2
```

1.7.3 Method 3: Manual Curl Commands

Generate and use curl commands for manual testing:

```
# Generate commands
./generate_curl_commands.sh -m "Your test message"

# Use generated files
./discovery_command.sh | jq
```

```
./send_message_command.sh | jq

# Or copy-paste the one-liner commands provided
```

1.8 Python Client Implementation

1.8.1 Complete Generic Client Code

```
#!/usr/bin/env python3
"""
Generic ServiceNow A2A Client
A configurable client for connecting to any ServiceNow Agent-to-Agent (A2A) endpoint.
"""

import asyncio
from uuid import uuid4
import httpx
import os
from pathlib import Path
from typing import Optional, Dict, Any
from dataclasses import dataclass

# Load environment variables from .env file if it exists
env_file = Path(__file__).parent / '.env'
if env_file.exists():
    with open(env_file) as f:
        for line in f:
            if line.strip() and not line.startswith('#'):
                key, value = line.strip().split('=', 1)
                os.environ[key] = value

from a2a.client import A2AClient
from a2a.types import (
    AgentCard,
    SendMessageRequest,
    MessageSendParams,
)


@dataclass
class ServiceNowConfig:
    """Configuration for ServiceNow A2A client"""
    instance: str  # e.g., "your-instance.service-now.com"
    agent_id: str  # ServiceNow agent ID
    api_key: str   # ServiceNow API key
```

```
    @property
    def base_url(self) -> str:
        """Base URL for A2A API"""
        return f"https://{self.instance}/api/sn_aia/a2a/id"

    @property
    def discovery_url(self) -> str:
        """Agent discovery URL"""
        return f"{self.base_url}/{self.agent_id}/well_known/agent_json"

    @classmethod
    def from_env(cls) -> 'ServiceNowConfig':
        """Create configuration from environment variables"""
        instance = os.getenv("SERVICENOW_INSTANCE")
        agent_id = os.getenv("SERVICENOW_AGENT_ID")
        api_key = os.getenv("SERVICENOW_TOKEN")

        if not instance:
            raise ValueError("SERVICENOW_INSTANCE environment variable is required")
        if not agent_id:
            raise ValueError("SERVICENOW_AGENT_ID environment variable is required")
        if not api_key:
            raise ValueError("SERVICENOW_TOKEN environment variable is required")

        return cls(instance=instance, agent_id=agent_id, api_key=api_key)


class ServiceNowA2AClient:
    """Generic ServiceNow A2A Client"""

    def __init__(self, instance: str = None, agent_id: str = None, api_key: str = None):
        """Initialize the ServiceNow A2A client"""
        if instance and agent_id and api_key:
            self.config = ServiceNowConfig(instance=instance, agent_id=agent_id,
        api_key=api_key)
        else:
            self.config = ServiceNowConfig.from_env()

        self.agent_card: Optional[AgentCard] = None
        self.a2a_client: Optional[A2AClient] = None

    async def fetch_agent_card(self, httpx_client: httpx.AsyncClient) -> AgentCard:
        """Fetch agent card from ServiceNow discovery endpoint"""
        try:
            print(f"🔍 Discovering agent at: {self.config.instance}")

            headers = {
                'content-type': 'application/json',
                'x-sn-apikey': self.config.api_key,
```

```
                'user-agent': 'ServiceNow-A2A-Client/1.0'
            }

            response = await httpx_client.get(self.config.discovery_url, headers=headers)
            response.raise_for_status()

            agent_card_data = response.json()
            agent_card = AgentCard(**agent_card_data)

            print("✅ Agent discovered successfully!")
            print(f"📝 Name: {agent_card.name}")

            self.agent_card = agent_card
            return agent_card

        except Exception as e:
            print(f"❌ Error fetching agent card: {e}")
            raise RuntimeError(f"Failed to fetch agent card: {e}")

    def initialize_client(self, httpx_client: httpx.AsyncClient) -> A2AClient:
        """Initialize A2A client with the discovered agent card"""
        if not self.agent_card:
            raise RuntimeError("Agent card not fetched. Call fetch_agent_card() first.")

        client = A2AClient(httpx_client=httpx_client, agent_card=self.agent_card)
        print("✅ A2A Client initialized")

        self.a2a_client = client
        return client

    async def send_message(self, message_text: str, context_id: str = None, task_id: str =
        None) -> Dict[str, Any]:
        """Send a message to the ServiceNow agent"""
        if not self.a2a_client:
            raise RuntimeError("A2A client not initialized. Call initialize_client()
        first.")

        print(f"💬 Sending message: {message_text}")

        request_id = uuid4().hex
        message_body = {
            'message': {
                'role': 'user',
                'parts': [{'kind': 'text', 'text': message_text}],
                'messageId': uuid4().hex,
                'kind': 'message'  # SDK expects 'message'
            },
            'context': {'contextId': context_id, 'taskId': task_id},
            'metadata': {},
            'pushNotificationUrl':
        f"http://localhost:8080/a2a/callback/{self.config.agent_id}/{request_id}",
```

```
            'id': request_id
        }

        request = SendMessageRequest(id=str(uuid4()),
        params=MessageSendParams(**message_body))

        try:
            response = await self.a2a_client.send_message(request)
            print("✅ Message sent successfully!")
            return response.model_dump(mode='json', exclude_none=True)
        except Exception as e:
            print(f"❌ Error sending message: {e}")
            raise

    async def connect_and_send(self, message_text: str, context_id: str = None, task_id: str
        = None) -> Dict[str, Any]:
        """Convenience method to connect to agent and send a message in one call"""
        timeout = httpx.Timeout(30.0, connect=10.0)

        async with httpx.AsyncClient(timeout=timeout) as httpx_client:
            await self.fetch_agent_card(httpx_client)
            self.initialize_client(httpx_client)
            return await self.send_message(message_text, context_id, task_id)


async def main():
    """Example usage of the generic ServiceNow A2A client"""
    print("=== Generic ServiceNow A2A Client ===")

    try:
        client = ServiceNowA2AClient()

        print(f"🖥️ Instance: {client.config.instance}")
        print(f"🤖 Agent ID: {client.config.agent_id}")

        # Send a message to the agent
        message = "What can you help me with?"
        response = await client.connect_and_send(message)

        print("\n✅ A2A Client completed successfully!")

    except ValueError as e:
        print(f"❌ Configuration Error: {e}")
        print("\nRequired environment variables:")
        print("  SERVICENOW_INSTANCE=your-instance.service-now.com")
        print("  SERVICENOW_AGENT_ID=your-agent-id")
        print("  SERVICENOW_TOKEN=your-api-key")
    except Exception as e:
        print(f"❌ A2A Client failed: {e}")
```

```
if __name__ == "__main__":
    asyncio.run(main())
```

1.9 Curl Testing Scripts

1.9.1 Automated Testing Script

```
#!/bin/bash
# ServiceNow A2A Agent Test Script
# This script generates and executes curl commands to test any ServiceNow agent

set -e

# Color codes for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m'

print_status() { echo -e "${BLUE}[INFO]${NC} $1"; }
print_success() { echo -e "${GREEN}[SUCCESS]${NC} $1"; }
print_error() { echo -e "${RED}[ERROR]${NC} $1"; }

# Load configuration from .env file if it exists
if [ -f .env ]; then
    print_status "Loading configuration from .env file..."
    export $(grep -v '^#' .env | xargs)
fi

# Configuration variables
SERVICENOW_INSTANCE=${SERVICENOW_INSTANCE:-""}
SERVICENOW_AGENT_ID=${SERVICENOW_AGENT_ID:-""}
SERVICENOW_TOKEN=${SERVICENOW_TOKEN:-""}

# Parse command line arguments
CUSTOM_MESSAGE=""
DISCOVERY_ONLY=false
VERBOSE=false

while [[ $# -gt 0 ]]; do
    case $1 in
        -i|--instance) SERVICENOW_INSTANCE="$2"; shift 2 ;;
        -a|--agent-id) SERVICENOW_AGENT_ID="$2"; shift 2 ;;
        -t|--token) SERVICENOW_TOKEN="$2"; shift 2 ;;
```

```
        -m|--message) CUSTOM_MESSAGE="$2"; shift 2 ;;
        -d|--discovery-only) DISCOVERY_ONLY=true; shift ;;
        -v|--verbose) VERBOSE=true; shift ;;
        -h|--help)
            echo "Usage: $0 [OPTIONS]"
            echo "Test a ServiceNow A2A agent using curl commands"
            echo ""
            echo "Options:"
            echo "  -i, --instance INSTANCE    ServiceNow instance"
            echo "  -a, --agent-id AGENT_ID    Agent ID"
            echo "  -t, --token TOKEN          API token"
            echo "  -m, --message MESSAGE      Custom message"
            echo "  -d, --discovery-only       Only test discovery"
            echo "  -v, --verbose              Verbose output"
            echo "  -h, --help                 Show this help"
            exit 0 ;;
        *) print_error "Unknown option: $1"; exit 1 ;;
    esac
done

# Validate required configuration
if [ -z "$SERVICENOW_INSTANCE" ] || [ -z "$SERVICENOW_AGENT_ID" ] || [ -z
        "$SERVICENOW_TOKEN" ]; then
    print_error "Missing required configuration"
    echo "Required: SERVICENOW_INSTANCE, SERVICENOW_AGENT_ID, SERVICENOW_TOKEN"
    exit 1
fi

# Set default message
if [ -z "$CUSTOM_MESSAGE" ]; then
    CUSTOM_MESSAGE="Hello! This is a test message. What can you help me with?"
fi

# Build URLs
BASE_URL="https://${SERVICENOW_INSTANCE}/api/sn_aia/a2a/id"
DISCOVERY_URL="${BASE_URL}/${SERVICENOW_AGENT_ID}/well_known/agent_json"
INVOCATION_URL="https://${SERVICENOW_INSTANCE}/api/sn_aia/a2a/v1/agent/id/${SERVICENOW_AGENT_ID}"

print_status "ServiceNow A2A Agent Test"
echo "================================="
echo "Instance: $SERVICENOW_INSTANCE"
echo "Agent ID: $SERVICENOW_AGENT_ID"
echo "Message:  $CUSTOM_MESSAGE"
echo "================================="

# Test 1: Agent Discovery
print_status "Step 1: Testing agent discovery..."

DISCOVERY_RESPONSE=$(curl -s -w "\n%{http_code}" \
    -H "Content-Type: application/json" \
```

```
    -H "x-sn-apikey: $SERVICENOW_TOKEN" \
    -H "User-Agent: ServiceNow-A2A-Test/1.0" \
    "$DISCOVERY_URL")

HTTP_STATUS=$(echo "$DISCOVERY_RESPONSE" | tail -n 1)
DISCOVERY_BODY=$(echo "$DISCOVERY_RESPONSE" | sed '$d')

if [ "$HTTP_STATUS" = "200" ]; then
    print_success "Agent discovery successful!"

    AGENT_NAME=$(echo "$DISCOVERY_BODY" | python3 -c "import sys, json;
        data=json.load(sys.stdin); print(data.get('name', 'Unknown'))" 2>/dev/null || echo
        "Unknown")
    echo "  Agent Name: $AGENT_NAME"

    if [ "$VERBOSE" = true ]; then
        echo ""
        print_status "Full agent card response:"
        echo "$DISCOVERY_BODY" | python3 -m json.tool 2>/dev/null || echo "$DISCOVERY_BODY"
    fi
else
    print_error "Agent discovery failed! HTTP Status: $HTTP_STATUS"
    echo "Response: $DISCOVERY_BODY"
    exit 1
fi

# Exit if discovery-only mode
if [ "$DISCOVERY_ONLY" = true ]; then
    print_success "Discovery-only mode completed successfully!"
    exit 0
fi

# Test 2: Send Message
print_status "Step 2: Sending test message to agent..."

REQUEST_ID=$(uuidgen | tr '[:upper:]' '[:lower:]' | tr -d '-')
MESSAGE_ID=$(uuidgen | tr '[:upper:]' '[:lower:]' | tr -d '-')

JSON_PAYLOAD=$(cat <<EOF
{
  "jsonrpc": "2.0",
  "method": "message/send",
  "params": {
    "message": {
      "role": "user",
      "parts": [{"kind": "text", "text": "$CUSTOM_MESSAGE"}],
      "messageId": "$MESSAGE_ID",
      "kind": "user"
    },
    "context": {"contextId": null, "taskId": null},
    "metadata": {},
```

```
    "pushNotificationUrl": "http://localhost:8080/a2a/callback/test/$REQUEST_ID",
    "id": "$REQUEST_ID"
  },
  "id": "$REQUEST_ID"
}
EOF
)

MESSAGE_RESPONSE=$(curl -s -w "\n%{http_code}" \
    -X POST \
    -H "Content-Type: application/json" \
    -H "x-sn-apikey: $SERVICENOW_TOKEN" \
    -H "User-Agent: ServiceNow-A2A-Test/1.0" \
    -d "$JSON_PAYLOAD" \
    "$INVOCATION_URL")

HTTP_STATUS=$(echo "$MESSAGE_RESPONSE" | tail -n 1)
MESSAGE_BODY=$(echo "$MESSAGE_RESPONSE" | sed '$d')

if [ "$HTTP_STATUS" = "200" ]; then
    print_success "Message sent successfully!"

    # Extract agent response
    AGENT_RESPONSES=$(echo "$MESSAGE_BODY" | python3 -c "
import sys, json
try:
    data = json.load(sys.stdin)
    result = data.get('result', {})
    status = result.get('status', {})
    message = status.get('message', {})
    parts = message.get('parts', [])

    for i, part in enumerate(parts):
        if part.get('kind') == 'text':
            print(f'Response {i+1}: {part.get(\"text\", \"\")}')
except:
    print('Could not parse agent response')
" 2>/dev/null)

    if [ -n "$AGENT_RESPONSES" ]; then
        print_status "Agent Response:"
        echo "$AGENT_RESPONSES"
    fi

    if [ "$VERBOSE" = true ]; then
        echo ""
        print_status "Full response:"
        echo "$MESSAGE_BODY" | python3 -m json.tool 2>/dev/null || echo "$MESSAGE_BODY"
    fi
else
```

```
    print_error "Message sending failed! HTTP Status: $HTTP_STATUS"
    echo "Response: $MESSAGE_BODY"
    exit 1
fi

print_success "All tests completed successfully!"
```

1.10 Troubleshooting

1.10.1 Common Issues and Solutions

1.10.1.1 1. Authentication Errors (401 Unauthorized)

Problem: API key authentication fails

Solutions: - Verify your API key is correct and not expired - Ensure the API key has A2A permissions - Check that you're using the correct ServiceNow instance URL - Try regenerating the API key in ServiceNow

1.10.1.2 2. Agent Not Found (404 Not Found)

Problem: Agent discovery fails

Solutions: - Verify the agent ID is correct - Ensure the agent is published and active - Check the discovery URL format: `{instance}/api/sn_aia/a2a/id/{agent_id}/well_known/agent_json`

1.10.1.3 3. Connection Timeouts

Problem: Requests timeout or fail to connect

Solutions: - Check network connectivity to your ServiceNow instance - Verify the instance URL is accessible - Try increasing timeout values in the scripts - Check for firewall or proxy issues

1.10.1.4 4. Invalid JSON Response

Problem: Malformed JSON in responses

Solutions: - Check if the agent is properly configured - Verify the agent supports the A2A protocol - Try the discovery endpoint first to validate connectivity

1.10.1.5 5. Python Import Errors

Problem: Missing Python dependencies

Solutions:

```
# Install required packages
pip install httpx a2a-sdk

# Or create a virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install httpx a2a-sdk
```

1.10.2 Debugging Tips

1. **Use Verbose Mode:**

```
./test_servicenow_agent.sh -v
```

2. **Test Discovery Only:**

```
./test_servicenow_agent.sh --discovery-only
```

3. **Check Raw Responses:**

```
curl -v -H "x-sn-apikey: your-key" "your-discovery-url"
```

4. **Validate JSON:**

```
./discovery_command.sh | jq
```

1.11 Appendix

1.11.1 A. ServiceNow A2A Protocol Details

1.11.1.1 Discovery URL Pattern

```
https://{instance}/api/sn_aia/a2a/id/{agent_id}/well_known/agent_json
```

1.11.1.2 Invocation URL Pattern

```
https://{instance}/api/sn_aia/a2a/v1/agent/id/{agent_id}
```

1.11.1.3 Required Headers

```
Content-Type: application/json
x-sn-apikey: your-api-key
User-Agent: your-client-name
```

1.11.1.4 Message Format

```
{
  "jsonrpc": "2.0",
  "method": "message/send",
  "params": {
    "message": {
      "role": "user",
      "parts": [{"kind": "text", "text": "your message"}],
      "messageId": "unique-message-id",
      "kind": "user"
    },
    "context": {"contextId": null, "taskId": null},
    "metadata": {},
    "pushNotificationUrl": "your-callback-url",
    "id": "unique-request-id"
  },
  "id": "unique-request-id"
}
```

1.11.2 B. File Structure

```
servicenow-a2a-testing/
├── .env                          # Configuration file
├── .env.template                 # Configuration template
├── generic_a2a_client.py         # Python A2A client
├── test_servicenow_agent.sh      # Automated testing script
├── generate_curl_commands.sh     # Curl command generator
├── examples.py                   # Python usage examples
├── README.md                     # Documentation
└── requirements.txt              # Python dependencies
```

1.11.3 C. Environment Template

```
# ServiceNow A2A Client Configuration
# Copy this file to .env and fill in your values

# ServiceNow Instance (without https://)
SERVICENOW_INSTANCE=your-instance.service-now.com

# ServiceNow Agent ID
SERVICENOW_AGENT_ID=your-agent-id-here

# ServiceNow API Key
SERVICENOW_TOKEN=your-api-key-here
```

1.11.4 D. Requirements File

```
httpx>=0.24.0
a2a-sdk>=1.0.0
```

1.12 Support and Resources

1.12.1 ServiceNow Documentation

- ServiceNow A2A Documentation
- Agent Intelligence Platform

1.12.2 Community Resources

- ServiceNow Community Forums
- ServiceNow Developer Portal
- A2A SDK Documentation

1.12.3 Contact Information

For technical support with these testing tools, please refer to your ServiceNow administrator or the ServiceNow community resources.

Document Version: 1.0
Last Updated: September 2025