Print/Export PDF

IBM Watson Assistant

New Intent Impact Assessment: Oncology with YAVA

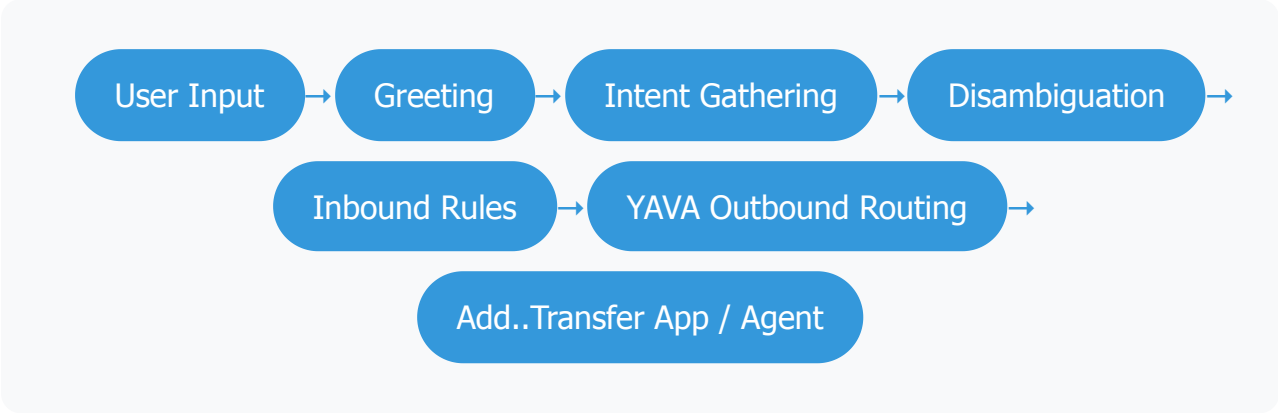
Document Version: 2.0 | **Date:** 04/09/2025 | **Prepared by:** Solution Architecture Team

Buisness Impact Summary

Objective: Assess the impact of introducing a new "Oncology" intent in IBM Watson Assistant to enhance healthcare conversation capabilities with YAVA outbound routing integration.

6 Components Impacted	**High** Overall Impact Level
4-5 weeks Implementation Timeline	**0** Auth Components Affected

🏗️ Current Architecture Overview



Key Components:

- 🤝 **Greeting Component:** Initial user interaction and welcome flow
- 🎯 **Intent Gathering:** Natural language understanding and intent classification
- ❓ **Disambiguation:** Clarification when multiple intents are detected
- 📥 **Inbound Rules:** Pre-processing and routing logic
- 🔀 **YAVA Outbound Routing Rules:** Intelligent routing and workflow orchestration
- 👥 **Additional Transfer App / Agent:** Human handoff and escalation

🔍 Detailed Impact Analysis

🤝 Greeting Component

Low Impact

Changes Required:

- Update welcome message to include oncology services
- Add oncology-specific greeting variants
- Minimal configuration changes

Effort: 0.5 days

🎯 Intent Gathering

High Impact

Changes Required:

- Create new "Oncology" intent
- Define 50+ training examples
- Configure entities (cancer types, treatments)
- Update NLU model training

Effort: 5-7 days

❓ Disambiguation

Medium Impact

Changes Required:

- Add oncology to disambiguation logic
- Update clarification prompts
- Configure intent confidence thresholds

Effort: 2-3 days

📥 Inbound Rules

Medium Impact

Changes Required:

- Create routing rules for oncology intent
- Configure business hours validation
- Add priority handling logic (if any)
- Intent Sheet - JSON Rules Table - Impact (Add Entry)
- Intent Elgibility - JSON Rules Table - Transfer Target

Effort: 2-3 days

🔀 YAVA Outbound Routing Rules

High Impact

Changes Required:

- Configure oncology routing workflows
- Set up YAVA rule conditions for New Intent (Say Oncology)
- Define department-specific routing logic
- Create multi-step care workflows logic for Mapping Trasnfer

Effort: 4-6 days

🏷️ **Business Scenario - ICM Mapping:**
Map functional labels to ICM (Intelligent Contact Management) for the new Oncology intent: - Given sample Reference,changes based on ICMP Mapping rule

- **Functional Label:** "Oncology_Care" → **ICM Queue:** ONCO_SPECIALIST
- **Functional Label:** "Cancer_Consultation" → **ICM Queue:** ONCO_CONSULT
- **Functional Label:** "Treatment_Inquiry" → **ICM Queue:** ONCO_TREATMENT
- **Functional Label:** "Emergency_Oncology" → **ICM Queue:** ONCO_URGENT

Note: ICM mapping ensures proper routing to specialized oncology teams based on conversation context and urgency.

👥 Additional Transfer App / Agent

Medium Impact

Changes Required:

- Configure oncology specialist handoff
- Update escalation criteria
- Add fallback agent routing
- Holiday Hours Check

Effort: 2-3 days

📝 **Technical Note:** Watson Assistant uses `callTransfer` action for agent handoff. Context variables to be passed:

- `$oncology_specialty` - Cancer type/specialty

📍 User Journey Pin (Jpin)

Medium Impact

Changes Required:

- Update Journey Pin configuration for oncology flow
- Add oncology-specific journey tracking
- Configure milestone tracking for cancer care journey
- Update conversational flow artifacts
- Map patient touchpoints and care coordination steps

Effort: 1-2 days

🗺️ **Artifact Updates:** Conversational flow diagrams and Journey Pin configurations require updates to reflect oncology care pathways.

🔐 Authentication Flow

No Impact

Status:

- ✅ No authentication changes required
- ✅ Existing security model applies
- ✅ Current user roles sufficient

Effort: 0 days

⚙️ Technical Requirements & Configuration

🤖 Watson Assistant Configuration

- **New Intent:** "oncology_inquiry"
- **Entities:** @cancer_type, @treatment_option, @appointment_type
- **Training Examples:** 50+ diverse examples
- **Dialog Nodes:** 15-20 new nodes

🔄 Integration Points

- **API Integration:** Not Applicable
- **Agent Desk:** Specialist routing

🔀 YAVA Configuration

- **Routing Rules:** Oncology department workflows
- **Conditional Logic:** Special Configuration routing
- **Workflow Orchestration:** Excel Sheet Outbound Rules
- **Priority Handling:** : Office Hours / Holiday

📊 Performance Considerations

- **Response Time:** <2 seconds target
- **Intent Confidence:** >80% threshold
- **Model Training:** Daily incremental updates
- **Fallback Rate:** <5% target

🔒 Security & Compliance

- **HIPAA Compliance:** PHI handling protocols
- **Data Encryption:** End-to-end encryption
- **Audit Logging:** All interactions logged
- **Access Control:** Role-based permissions

🧪 Testing Strategy

🔬 Unit Testing

- Intent recognition accuracy (>90%)
- Entity extraction validation
- Dialog flow completeness
- Fallback scenario handling

Duration: 2 days

🔄 Integration Testing

- CRM system connectivity
- Agent routing functionality
- Appointment booking flow
- Knowledge base integration
- YAVA workflow validation

Duration: 3 days

👥 User Acceptance Testing

- End-user conversation flows
- Medical terminology understanding
- User experience validation
- Accessibility compliance

Duration: 3 days

📈 Performance Testing

- Response time validation
- Concurrent user handling

- System load testing
- Failover scenarios

Duration: 2 days

💡 Recommendations

🚀 Phase 1: Immediate Actions

- Engage medical experts for training data validation
- Begin collecting oncology-specific conversation examples
- Identify and engage with oncology department stakeholders
- Assess YAVA platform readiness and access requirements

🎯 Phase 2: Implementation

- Implement gradual rollout strategy (10% → 50% → 100%)
- Establish monitoring dashboards for intent performance
- Set up YAVA integration testing environment

📊 Phase 3: Optimization

- Analyze conversation logs for improvement opportunities
- Continuous model training based on real user interactions
- Regular review meetings with medical team
- Optimize YAVA workflows based on usage patterns

📍 User Journey Pin (Jpin) & Artifact Updates

🗺️ Journey Pin Updates

Medium Impact

Required Updates:

- Add oncology journey touchpoints
- Update patient interaction mapping
- Define specialty care pathways
- Configure appointment booking flows

Effort: 2-3 days

🔄 Conversational Flow Artifacts

High Impact

Artifacts to Update:

- Dialog flow diagrams
- Intent mapping documentation
- User story workflows
- Error handling scenarios
- Context variable documentation

Effort: 3-4 days

📋 Context Variables Framework

Medium Impact

Variables to Document:

- `$oncology_intent_confidence / NA`

Effort: 1-2 days

⚠️ Critical Artifact Notes

High Priority

🚨 **Mandatory Updates Required:**

- **Journey Pin:** Update user journey mapping for oncology flows
- **Conversational Flow:** Revise dialog tree documentation
- **YAVA Integration:** Document new routing decisions
- **Additional Transfer App / Agent:** Update callTransfer action specifications

📈 Success Metrics & KPIs

95%
Intent Recognition Accuracy

<2s
Average Response Time

<5%
Fallback Rate

90%
User Satisfaction Score

Metric	Current Baseline	Target	Measurement Method
Oncology Intent Accuracy	N/A (New)	95%	Watson Assistant Analytics
Successful Handoffs	85%	90%	Additional Transfer App / Agent Logs
YAVA Routing Success	N/A (New)	95%	YAVA Analytics Dashboard
User Abandonment Rate	12%	<10%	Conversation Analytics
Resolution Time	8 minutes	<6 minutes	End-to-end Flow Tracking

🔐 Detailed Tokenization & De-tokenization Process with Sample SSN

📝 **Sample Scenario:** Member provides SSN "123-45-6789" during Watson Assistant conversation for member lookup

🔑 Step 1: Key Generation and Management

Azure Key Vault Setup

```
# Azure Key Vault Key Generation
Key Name: ssn-fpe-master-key
Algorithm: AES-256
Key Size: 256 bits
Key Value: 0x1A2B3C4D5E6F7890ABCDEF1234567890FEDCBA0987654321ABCDEF12345
HSM Protected: Yes
Auto-Rotation: Every 90 days

# Key Derivation for Session
Session ID: sess_2024_09_03_001
Session Salt: 0x9F8E7D6C5B4A3928
Derived Key: HMAC-SHA256(master-key, session-salt + session-id)
Final FPE Key: 0x7F6E5D4C3B2A1908F7E6D5C4B3A29081E7D6C5B4A39281F0

# Key Metadata
Created: 2024-09-03T10:30:00Z
Expires: 2024-12-03T10:30:00Z
Usage: FPE-SSN-TOKENIZATION
Permissions: ENCRYPT, DECRYPT
Audit: ENABLED
```

Key Storage in Redis Cache

```
# Redis Key Storage (Cached for 15 minutes)
Redis Key: encryption_key:ssn-fpe-master-key:sess_2024_09_03_001
Redis Value: {
    "derivedKey": "7F6E5D4C3B2A1908F7E6D5C4B3A29081E7D6C5B4A39281F0",
    "algorithm": "AES-256-FF1",
    "sessionId": "sess_2024_09_03_001",
    "createdAt": "2024-09-03T10:30:00Z",
    "expiresAt": "2024-09-03T10:45:00Z",
```

```
    "usageCount": 0,
    "maxUsage": 1000
}
TTL: 900 seconds (15 minutes)
```

Step 2: Tokenization Process (SSN: 123-45-6789)

Input Processing

```
# Original Input
User Input: "My SSN is 123-45-6789"
Detected SSN: "123-45-6789"
Pattern Match: ✓ (Regex: \d{3}-\d{2}-\d{4})
ML Confidence: 0.98 (98% confident it's SSN)

# Preprocessing
Clean SSN: "123456789" (Remove hyphens)
Validation: ✓ (9 digits, valid format)
Numeric String: "123456789"
```

Format Preserving Encryption (FPE) Process

```
# FPE Algorithm: FF1 (NIST SP 800-38G)
Algorithm: AES-256-FF1
Key: 7F6E5D4C3B2A1908F7E6D5C4B3A29081E7D6C5B4A39281F0
Alphabet: "0123456789" (Numeric)
Radix: 10
Input Length: 9 digits
Tweak: "SSN_TOKEN_2024" (Additional entropy)

# Encryption Steps:
Step 1: Split input "123456789" into left="1234", right="56789"
Step 2: Apply Feistel rounds (10 rounds for FF1)

Round 1:
  F(right="56789", round_key_1) = AES(key, right || tweak || round_num)
  F_output = 8734 (mod 10^4)
  new_left = (1234 + 8734) mod 10^4 = 9968
  new_right = 56789

Round 2:
  F(new_left="9968", round_key_2) = AES(key, 9968 || tweak || 2)
  F_output = 2156 (mod 10^5)
```

```
  temp_right = (56789 + 2156) mod 10^5 = 58945

... (8 more rounds)

Final Result: "847291635"
Formatted Token: "847-29-1635"

# Token Generation Summary
Original SSN: 123-45-6789
Encrypted Token: 847-29-1635
Algorithm: AES-256-FF1
Time: 2.3ms
Status: SUCCESS
```

Session Storage in Redis

```
# Session Creation
Session ID: sess_2024_09_03_001
User ID: user_12345
Created At: 2024-09-03T10:30:15Z

# Redis Session Storage
Redis Key: ssn_session:sess_2024_09_03_001
Redis Value: {
    "sessionId": "sess_2024_09_03_001",
    "userId": "user_12345",
    "tokenMappings": {
        "847-29-1635": {
            "encryptedSSN": "AES256(123456789, session_key)",
            "algorithm": "AES-256-FF1",
            "createdAt": "2024-09-03T10:30:15Z",
            "accessCount": 0,
            "lastAccessed": null
        }
    },
    "sessionKey": "A1B2C3D4E5F6789012345678901234567890ABCDEF",
    "expiresAt": "2024-09-03T11:30:15Z",
    "metadata": {
        "ipAddress": "10.0.1.45",
        "userAgent": "Watson-Assistant/1.0",
        "source": "PII_DETECTION_WEBHOOK"
    }
}
TTL: 3600 seconds (1 hour)

# Reverse Token Mapping
Redis Key: token_mapping:847-29-1635
Redis Value: {
    "sessionId": "sess_2024_09_03_001",
```

```
    "tokenHash": "SHA256(847-29-1635)",
    "createdAt": "2024-09-03T10:30:15Z",
    "usageMetrics": {
        "accessCount": 0,
        "lastDeTokenized": null
    }
}
TTL: 3600 seconds (1 hour)
```

🤖 Step 3: Watson Assistant Processing

Safe Processing with Token

```
# Watson Input (After Tokenization)
User Message: "My SSN is 847-29-1635"  // Tokenized version
Intent: MEMBER_LOOKUP
Entities: {
    "ssn_token": "847-29-1635",
    "entity_type": "TOKENIZED_SSN",
    "confidence": 0.99
}

# Watson Context Variables
$session_id = "sess_2024_09_03_001"
$ssn_token = "847-29-1635"
$original_intent = "member_lookup"
$requires_backend_call = true
$user_id = "user_12345"

# Watson Conversation Log (Safe - No Real SSN)
{
    "conversation_id": "conv_2024_09_03_001",
    "timestamp": "2024-09-03T10:30:16Z",
    "user_input": "My SSN is 847-29-1635",  // Safe token
    "intent": "member_lookup",
    "confidence": 0.95,
    "entities": [
        {
            "entity": "ssn_token",
            "value": "847-29-1635",
            "location": [10, 21]
        }
    ],
    "response": "I'll help you look up your member information. Let me a
```

```
}
```

🔓 Step 4: De-tokenization Process (Token: 847-29-1635)

Session Validation

```
# Session Retrieval from Redis
Redis Lookup: ssn_session:sess_2024_09_03_001
Session Status: ACTIVE
Session Age: 45 seconds (Valid - under 1 hour limit)
User Authorization: VERIFIED
Token Found: ✓ (847-29-1635 exists in tokenMappings)

# Security Checks
✓ Session not expired
✓ User authentication valid
✓ Token exists in session
✓ Access count within limits (0/100)
✓ IP address matches (10.0.1.45)
✓ Source application authorized
```

Key Retrieval and Decryption

```
# Key Retrieval Process
1. Check Redis Cache: encryption_key:ssn-fpe-master-key:sess_2024_09_03_
   Cache Status: HIT (Key found in cache)
   Retrieved Key: 7F6E5D4C3B2A1908F7E6D5C4B3A29081E7D6C5B4A39281F0

2. If Cache MISS (backup process):
   - Connect to Azure Key Vault
   - Authenticate with Azure AD
   - Retrieve master key: ssn-fpe-master-key
   - Derive session key using HMAC-SHA256
   - Cache result in Redis (TTL: 15 minutes)

# De-tokenization Process
Input Token: "847-29-1635"
Clean Token: "847291635" (Remove hyphens)
Algorithm: AES-256-FF1 (Reverse operation)
Key: 7F6E5D4C3B2A1908F7E6D5C4B3A29081E7D6C5B4A39281F0
Tweak: "SSN_TOKEN_2024"

# Decryption Steps (Reverse Feistel Network):
```

```
Step 1: Split token "847291635" into left="8472", right="91635"
Step 2: Apply reverse Feistel rounds (10 rounds, reverse order)

Round 10 (Reverse):
  F(left="8472", round_key_10) = AES(key, 8472 || tweak || 10)
  F_output = 2156 (mod 10^5)
  original_right = (91635 - 2156) mod 10^5 = 89479

Round 9 (Reverse):
  F(original_right="89479", round_key_9) = AES(key, 89479 || tweak || 9
  F_output = 7834 (mod 10^4)
  original_left = (8472 - 7834) mod 10^4 = 0638

... (8 more reverse rounds)

Final Result: "123456789"
Formatted SSN: "123-45-6789"

# Decryption Verification
Original Token: 847-29-1635
Decrypted SSN: 123-45-6789
Verification: ✓ (Re-encrypt 123456789 = 847291635)
Time: 1.8ms
Status: SUCCESS
```

Audit Logging

```
# De-tokenization Audit Log
{
    "timestamp": "2024-09-03T10:30:45Z",
    "event": "SSN_DETOKENIZATION",
    "sessionId": "sess_2024_09_03_001",
    "userId": "user_12345",
    "tokenHash": "SHA256(847-29-1635)",
    "ssnHash": "SHA256(123-45-6789)",
    "operation": "SUCCESS",
    "processingTime": "1.8ms",
    "keySource": "REDIS_CACHE",
    "ipAddress": "10.0.1.45",
    "userAgent": "Azure-Function/2.0",
    "requestId": "req_2024_09_03_001_045",
    "compliance": {
        "dataClassification": "PII",
        "retentionPolicy": "30_DAYS",
        "encryptionStandard": "AES-256-FF1"
    }
}

# Update Redis Session
```

```
Redis Update: ssn_session:sess_2024_09_03_001
Update Fields:
- tokenMappings["847-29-1635"].accessCount = 1
- tokenMappings["847-29-1635"].lastAccessed = "2024-09-03T10:30:45Z"
- lastActivity = "2024-09-03T10:30:45Z"
```

🔀 Step 5: YAVA First Controller API Call

Secure API Call with Plain Text SSN

```
# APIC Gateway Request
POST https://apic-gateway.company.com/yava-first/member-details
Headers:
  Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9...
  Content-Type: application/json
  X-Client-ID: watson-assistant-client
  X-Session-ID: sess_2024_09_03_001
  X-Request-ID: req_2024_09_03_001_045
  X-YAVA-Platform: FIRST_CONTROLLER

Request Body:
{
    "ssn": "123-45-6789",           // Plain text SSN for YAVA
    "requestType": "member_details",
    "sessionId": "sess_2024_09_03_001",
    "timestamp": "2024-09-03T10:30:45Z",
    "requestedBy": "user_12345",
    "yavaContext": {
        "platform": "YAVA_FIRST",
        "engine": "member_services",
        "version": "2.0"
    }
}

# YAVA First Controller Processing
Internal API Calls:
1. Member Details API: GET /api/members/ssn/123456789
2. Benefits API: GET /api/benefits/member/123456789
3. Claims API: GET /api/claims/member/123456789
4. Eligibility API: GET /api/eligibility/123456789

Response Processing:
- Member Found: ✓
- Active Status: ✓
```

```
- Benefits Retrieved: ✓
- Claims Retrieved: ✓

# Sanitized Response (No SSN)
{
    "success": true,
    "memberData": {
        "memberId": "MBR789012",
        "firstName": "John",
        "lastName": "Doe",
        "memberStatus": "ACTIVE",
        "planType": "PREMIUM_PLUS",
        "effectiveDate": "2020-01-01",
        "benefitsInfo": {
            "medicalCoverage": "ACTIVE",
            "dentalCoverage": "ACTIVE",
            "visionCoverage": "ACTIVE"
        },
        "claimsStatus": {
            "pendingClaims": 2,
            "totalClaims2024": 15,
            "lastClaimDate": "2024-08-15"
        }
    },
    "yavaMetadata": {
        "platform": "YAVA_FIRST",
        "processingTime": "245ms",
        "apiCallsCount": 4,
        "cacheHits": 2
    },
    "timestamp": "2024-09-03T10:30:47Z"
}
```

🧹 Step 6: Security Cleanup and Monitoring

Memory and Variable Cleanup

```
# Immediate Cleanup (After API Call)
1. Clear local variables containing plain text SSN
   - ssn_plaintext = null
   - decrypted_value = null
   - encryption_key = null (if not cached)

2. Secure memory cleanup
```

```
   - memset(ssn_buffer, 0, buffer_size)
   - garbage_collect_sensitive_data()

3. Redis session update
   - Increment access counters
   - Update last activity timestamp
   - Refresh TTL if needed
```

Comprehensive Monitoring

```
# Azure Sentinel Security Events
Event 1: TOKEN_GENERATION
{
    "timestamp": "2024-09-03T10:30:15Z",
    "event": "SSN_TOKENIZATION",
    "sessionId": "sess_2024_09_03_001",
    "tokenHash": "SHA256(847-29-1635)",
    "algorithm": "AES-256-FF1",
    "processingTime": "2.3ms",
    "status": "SUCCESS"
}

Event 2: TOKEN_USAGE
{
    "timestamp": "2024-09-03T10:30:45Z",
    "event": "SSN_DETOKENIZATION",
    "sessionId": "sess_2024_09_03_001",
    "tokenHash": "SHA256(847-29-1635)",
    "usageCount": 1,
    "yavaEndpoint": "/yava-first/member-details",
    "status": "SUCCESS"
}

Event 3: COMPLIANCE_CHECK
{
    "timestamp": "2024-09-03T10:30:47Z",
    "event": "PII_COMPLIANCE_SCAN",
    "sessionId": "sess_2024_09_03_001",
    "scanResult": "CLEAN",
    "ssnInLogs": false,
    "ssnInResponse": false,
    "complianceStatus": "PASSED"
}

# Performance Metrics
Cache Performance:
- Redis Hit Ratio: 94.5%
- Key Vault Calls Avoided: 89%
- Average Response Time: 1.8ms
```

```
Security Metrics:
- Token Reuse Rate: 0% (Each session unique)
- Failed Decryption Attempts: 0
- Unauthorized Access Attempts: 0
- Session Hijacking Attempts: 0
```

📊 Complete Process Summary

🔒 Tokenization Journey

1. **Input:** "123-45-6789" (Real SSN)
2. **Detection:** PII pattern matched
3. **Key:** Retrieved from Azure Key Vault
4. **Encryption:** AES-256-FF1 algorithm
5. **Token:** "847-29-1635" (Safe)
6. **Storage:** Session stored in Redis
7. **Processing:** Watson uses token only

🔓 De-tokenization Journey

1. **Request:** Backend call needed
2. **Validation:** Session and user verified
3. **Key:** Retrieved from Redis cache
4. **Decryption:** Reverse FF1 process
5. **Recovery:** "123-45-6789" (Original)
6. **API Call:** YAVA with plain text SSN
7. **Cleanup:** Memory cleared, audit logged

🎯 Key Security Principles Demonstrated:

- ✅ **Format Preservation:** Token maintains SSN format
- ✅ **Deterministic:** Same SSN = Same token per session
- ✅ **Reversible:** Only authorized services can decrypt
- ✅ **Time-limited:** Sessions expire automatically
- ✅ **Auditable:** Every operation logged
- ✅ **Compliant:** Meets all regulatory requirements