Enterprise system integration architecture - Claude

claude.ai/chat/ba53fa8b-f8d9-4d73-a5a1-ef048c0d9e4e

I

how to integrate pc with an external system. take an example of mvr application with asyncronous outbound and syncronous inbound. take any real life example for triggers and business usecase and write me end to end

I'll walk you through a comprehensive integration scenario using a Motor Vehicle Records (MVR) application that needs to connect with external systems. Let me use a real-world example of a car rental company integrating with government DMV systems and insurance providers.

Business Use Case: TravelWheels Car Rental Company

TravelWheels needs to verify customer driving records and insurance status before approving rentals. They must check MVR data from state DMVs and validate insurance coverage with external providers.

Integration Architecture Overview

The system uses:

- **Asynchronous Outbound**: Batch processing of driver verifications overnight

- **Synchronous Inbound**: Real-time verification requests from rental kiosks/mobile apps

Detailed Implementation

1. System Components

Internal Systems:

- Rental Management System (RMS)
- Customer Database
- Risk Assessment Engine
- Notification Service

External Systems:

- State DMV APIs
- Insurance Verification Services (IVS)
- Credit Bureau Services

2. Asynchronous Outbound Integration

Business Trigger: Every night at 2 AM, the system processes pending reservations for the next 7 days to pre-verify customer driving records.

Implementation Flow:

The batch job initiates from the scheduler, pulling all upcoming reservations from the database. For each customer, the system creates an MVR request message containing driver's license number, state of issuance, and date of birth. These messages are placed in an outbound queue (using Apache Kafka or RabbitMQ).

A message processor picks up requests from the queue and groups them by state, since different DMVs have different APIs. The processor transforms each request into the specific format required by each state's DMV system - some use SOAP, others REST, and some still use file-based transfers.

For file-based DMVs, the system generates CSV or XML files and uploads them to designated SFTP servers. For API-based DMVs, the system makes HTTP calls with proper authentication tokens. Each request is logged with a correlation ID for tracking.

The external DMV systems process these requests on their schedule (typically 4-6 hours). They send responses back through various channels - webhook callbacks, file drops, or requiring polling of result endpoints.

Response Handling:

When responses arrive, a listener service captures them and matches them to original requests using correlation IDs. The MVR data is parsed and stored in the local database with fields like:

- License status (valid/suspended/expired)
- Violation history (last 3 years)
- DUI records
- Point accumulation
- License class and endorsements

The Risk Assessment Engine evaluates each record against rental policies. Customers with suspended licenses or recent DUIs are flagged as high-risk. The system sends email notifications to customers about their verification status and to branch managers about high-risk reservations.

3. Synchronous Inbound Integration

Business Trigger: A walk-in customer at a rental kiosk wants to rent a vehicle immediately.

Implementation Flow:

The customer enters their information at the kiosk or mobile app. The front-end application makes a REST API call to the integration middleware:

```
POST /api/v1/verify-driver
{
  "license_number": "D123456789",
  "state": "CA",
  "dob": "1985-03-15",
  "insurance_policy": "POL-789456",
  "rental_duration": 3
}
```

The middleware validates the request and implements circuit breaker patterns to handle external service failures. It makes parallel calls to both the DMV instant verification service and the insurance verification service, with a timeout of 10 seconds each.

For the DMV check, the system uses the state's real-time API (if available) or falls back to cached data if the record was checked within the last 30 days. The insurance verification involves calling the insurance provider's API to confirm active coverage and policy limits.

Synchronous Response Orchestration:

The middleware aggregates responses from both services. If either service times out, the system checks if cached data exists that's less than 7 days old. If no valid data is available, it returns a "manual verification required" status.

The aggregated response is sent back to the kiosk:

json

```json
{
  "verification_id": "VER-2024-789456",
  "driver_status": "APPROVED",
  "license_valid": true,
  "insurance_valid": true,
  "risk_score": 2,
  "restrictions": ["require_additional_deposit"],
  "cached_data_used": false,
  "timestamp": "2024-11-28T10:30:00Z"
}
```

4. Error Handling and Resilience

Retry Mechanisms:

- Exponential backoff for failed API calls (2s, 4s, 8s, up to 3 attempts)
- Dead letter queues for messages that fail after all retries
- Manual intervention queue for records requiring human review

Fallback Strategies:

- Use cached MVR data if less than 30 days old for non-critical rentals
- Allow provisional approval with higher deposits for low-risk customers
- Route to manual verification desk for high-value rentals

5. Data Transformation Layer

The system maintains transformation mappings for different state formats. For example, California DMV returns violations as codes (VC-23152 for DUI), while Texas uses descriptive text. The transformation layer standardizes these into a common internal format.

6. Security Implementation

Authentication & Authorization:

- OAuth 2.0 tokens for API access
- Certificate-based authentication for SFTP connections
- API keys stored in secure vaults (HashiCorp Vault)
- Role-based access control for internal users

Data Protection:

- TLS 1.3 for all API communications
- PGP encryption for file transfers
- PII data encryption at rest using AES-256
- Audit logging of all data access

7. Monitoring and Observability

Key Metrics Tracked:

- API response times (target: <2s for sync, <6h for async)
- Success rates by external provider
- Cache hit ratios
- Queue depths and processing delays
- Error rates by error type

Alerting Rules:

- Alert if DMV API success rate drops below 95%
- Page on-call if synchronous verification is down
- Notify if batch processing exceeds 8-hour SLA
- Alert on unusual spike in high-risk drivers

8. Business Rules Engine Integration

The system applies configurable business rules:

- Customers with >2 moving violations in 3 years require manager approval
- International licenses need additional documentation
- Drivers under 25 with violations pay 50% higher deposits
- Corporate accounts bypass certain checks based on contracts

9. Reporting and Analytics

The integration feeds a data warehouse for analytics:

- Daily reports on verification success rates
- Monthly analysis of risk scores vs actual claims
- Quarterly review of external service performance
- Annual audit reports for compliance

10. Disaster Recovery

Backup Strategies:

- Real-time replication of verification data to DR site
- Daily backups of configuration and mapping tables
- Ability to switch to DR endpoints within 15 minutes
- Manual verification procedures if all systems fail

This integration architecture ensures TravelWheels can efficiently verify drivers while maintaining system reliability and customer experience. The combination of batch processing for known reservations and real-time verification for walk-ins optimizes both system resources and customer convenience.

I

completely wrong.. what is the limit on number of files and size i can attach to each project in claude max plan

I apologize for the misunderstanding about your PC (PeopleSoft Campus) integration question. Let me answer your current question about Claude's file attachment limits.

For Claude's file attachment capabilities on the max plan (Claude Pro), here are the current limits:

Per Message:

- Up to 5 files can be attached per message
- Each file can be up to 30 MB in size
- Supported file types include: text files, PDFs, CSVs, JSON, code files, and various document formats

Per Project:

- Projects can contain up to 100 files total
- The cumulative size limit for all files in a project is approximately 200 MB
- Files added to a project persist across conversations within that project

Important Notes:

- These limits apply to the Claude Pro (paid) plan
- The free plan has more restrictive limits
- Very large files may take longer to process
- Claude can read and analyze the contents of uploaded files directly

For the most up-to-date information about specific plan limits and features, I'd recommend checking https://support.claude.com as these limits may change over time.

Regarding your original question about PeopleSoft Campus integration with MVR applications - would you like me to provide a proper explanation of PC (PeopleSoft Campus) integration patterns with asynchronous outbound and synchronous inbound messaging, using real-world examples from higher education institutions?