MODAPTO [101091996]: Modular Manufacturing and Distributed Control via Interoperable Digital Twins

11.2.1 Designing connector architecture and API specifications

2025-08-11

Designing a custom connector for the MODAPTO Orchestrator begins with a thorough analysis of the target system's architecture and API specifications. This analysis forms the foundation for creating a connector that can effectively bridge between the Orchestrator's standardized interface and the target system's specific requirements.

Requirements Analysis

Before designing a connector, it's essential to gather comprehensive information about the target system:

1. **Connection Method**: Determine how the system exposes its functionality (REST, SOAP, message queue, database, proprietary protocol, etc.)
2. **Authentication Requirements**: Identify what authentication mechanisms the system requires (API keys, OAuth, basic authentication, certificates, etc.)
3. **Data Formats**: Analyze the input and output data structures and formats (JSON, XML, CSV, binary, etc.)
4. **Operation Patterns**: Understand whether operations are synchronous or asynchronous, and how to handle each pattern
5. **Error Handling**: Identify how the system reports errors and what recovery mechanisms are available
6. **Rate Limits and Constraints**: Document any limitations on request frequency, payload size, or other constraints

Connector Architecture Design

Based on the requirements analysis, the connector architecture should be designed to address several key aspects:

1. **Connection Management**:
   - How the connector establishes and maintains connections to the target system
   - Connection pooling or reuse strategies if applicable
   - Timeout and retry policies
2. **Request Formatting**:
   - Transformation of Orchestrator inputs to target system format
   - Construction of appropriate headers, authentication tokens, etc.
   - Handling of different data types and structures
3. **Response Processing**:
   - Parsing and validation of system responses
   - Error detection and classification
   - Mapping between system-specific and standardized error codes
4. **State Management**:
   - Tracking of request/response pairs for asynchronous operations
   - Management of session information if required
   - Caching strategies for improving performance

API Specification Development

Once the architecture is defined, formal API specifications must be developed to document how the Orchestrator will interact with the connector:

1. **Input Parameter Specification**:
    - Define the parameters required during connector initialization
    - Specify the parameters needed for each operation execution
    - Document data types, validation rules, and default values
2. **Output Structure Definition**:
    - Define the structure and format of successful operation results
    - Specify error response formats and error codes
    - Document any metadata included in responses
3. **Operation Lifecycle**:
    - Define how operations are initialized, executed, and terminated
    - Specify any stateful behavior across multiple operations
    - Document any dependencies between operations

These specifications should be documented using standard formats such as OpenAPI (Swagger) for REST interfaces or WSDL for SOAP interfaces, ensuring clear understanding of the connector's capabilities and requirements.

References

1. MODAPTO Consortium. (2023). Orchestrator User Manual. MODAPTO Project Documentation.
2. OpenAPI Initiative. (2023). OpenAPI Specification. https://www.openapis.org/
3. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.
4. Richardson, L. (2018). RESTful Web APIs. O'Reilly Media.