Interface Design Description

Acme Widget Platform

Document Number: AWP-IDD-2025-001
Version: 2.0
January 20, 2025

Prepared by: Acme Engineering Team

Table of Contents

List of Figures

List of Tables

1 Scope

1.1 Identification

System Name	Acme Widget Platform (AWP)
CSCI Name	Widget Interface Services (WIS)
Contract Number	FA8750-24-C-0042
CDRL Sequence	A009
Document Identifier	AWP-IDD-2025-001

This Interface Design Description (IDD) identifies and describes the interfaces for the Acme Widget Platform (AWP) Computer Software Configuration Item (CSCI). This document is prepared in accordance with DI-IPSC-81436A[1] and conforms to the interface documentation requirements of MIL-STD-498[2].

1.2 System Overview

The Acme Widget Platform provides a comprehensive suite of services for creating, managing, and monitoring widgets across distributed environments. The system exposes four primary interfaces, shown in Figure 1:

- **IF-REST** — RESTful API for widget CRUD operations
- **IF-WS** — WebSocket service for real-time notifications
- **IF-DB** — PostgreSQL database interface for persistence
- **IF-MQ** — RabbitMQ message queue for asynchronous processing

These interfaces collectively support widget lifecycle management, event-driven notifications, data persistence, and inter-service communication.

1.3 Document Overview

This IDD is organized according to DI-IPSC-81436A:

a) **Section 1 — Scope:** System identification and overview.
b) **Section 2 — Referenced Documents:** Standards and references.
c) **Section 3 — Interface Design:** Detailed interface descriptions.
d) **Section 4 — Requirements Traceability:** Mapping to SRS requirements.
e) **Section 5 — Notes:** Glossary, acronyms, and assumptions.

[1] DI-IPSC-81436A is the Data Item Description for Interface Design Descriptions under MIL-STD-498.
[2] MIL-STD-498: Software Development and Documentation, Department of Defense Standard.

2 Referenced Documents

The following documents are referenced in this IDD:

a) **MIL-STD-498** — *Software Development and Documentation* (1994)
b) **DI-IPSC-81436A** — *Interface Design Description (IDD)* (1994)
c) **RFC 7231** — *HTTP/1.1 Semantics and Content* (2014)
d) **RFC 6455** — *The WebSocket Protocol* (2011)
e) **RFC 7807** — *Problem Details for HTTP APIs* (2016)
f) **AMQP 0-9-1** — *Advanced Message Queuing Protocol* (2008)
g) **ISO 8601** — *Date and Time Format* (2019)
h) **AWP-SRS-2025-001** — *Acme Widget Platform Software Requirements Specification* (2025)

3 Interface Design

3.1 Interface Identification and Diagrams

Table 1 provides an overview of all interfaces in the Acme Widget Platform.

Interface ID	Description	Protocol	Direction
IF-REST	Widget CRUD API	HTTPS	Bidirectional
IF-WS	Real-time notifications	WSS	Server → Client
IF-DB	Data persistence layer	TCP/SQL	Internal
IF-MQ	Async message processing	AMQP	Bidirectional

Table 1: Interface Summary

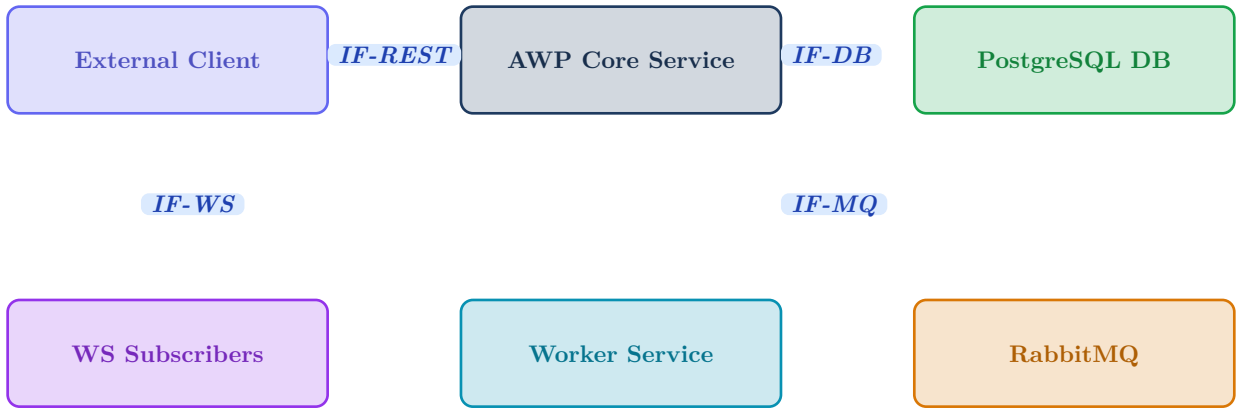

Figure 1: Interface Context Diagram

3.2 IF-REST: Widget REST API

3.2.1 Overview

The IF-REST interface provides a RESTful[1] API for widget lifecycle management. All operations conform to HTTP/1.1[RFC7231] semantics with JSON request and response bodies.

Authentication Required
All IF-REST endpoints require a valid API key in the `Authorization` header. Keys support scope-based access control.

3.2.2 Data Elements

Field	Type	Description	Constraint
widget_id	string	UUID v4 identifier	Read-only
name	string	Display name (1-255 chars)	Required
status	enum	active | paused | archived	Default: active
config	object	Widget configuration map	Optional
created_at	datetime	ISO 8601 creation timestamp	Read-only
updated_at	datetime	ISO 8601 last-modified timestamp	Read-only
version	integer	Optimistic lock version	Auto-increment

Table 2: IF-REST Widget Resource Data Elements

3.2.3 Request/Response Examples

Create Widget Request:

```
{
  "name": "Weather Dashboard",
  "config": {
    "refresh_interval": 300,
    "theme": "dark",
    "data_source": "api.weather.gov"
  }
}
```

Success Response (201 Created):

```
{
  "widget_id": "wgt_7f3a2b1c-9d4e-4a5f-b6c8-1e2d3f4a5b6c",
  "name": "Weather Dashboard",
  "status": "active",
  "version": 1,
  "created_at": "2025-01-15T10:30:00Z"
}
```

3.2.4 Requirements

IDD-REST-001 The **IF-REST** interface shall accept and return JSON payloads conforming to RFC 7159. Shall

IDD-REST-002 The **IF-REST** interface shall enforce rate limiting of 1000 requests per minute per API key. Shall

IDD-REST-003 The **IF-REST** interface should support conditional requests via `ETag` and `If-Match` headers for optimistic concurrency control. Should

IDD-REST-004 The **IF-REST** interface may support batch operations for up to 100 widgets per request. May

The service level agreement specifies 99.9% uptime, corresponding to a maximum of[3] 8.77 hours of annual downtime.

[3]Calculated as: $365.25 \times 24 \times 0.001 = 8.766$ hours of allowable downtime per year.

3.3 IF-WS: WebSocket Notification Service

3.3.1 Overview

The IF-WS interface provides real-time event delivery to subscribed clients via the WebSocket protocol (RFC 6455). Connections are upgraded from HTTPS and authenticated using the same API key mechanism as IF-REST.

3.3.2 Event Matrix

Event Type	Trigger	Scope	Priority	Payload Fields	Latency SLA	Retry
widget.created	POST /widgets	Project	Normal	widget_id, name, status, created_at, actor_id	< 500ms	No
widget.updated	PATCH /widgets/:id	Project	Normal	widget_id, changed_fields, version, actor_id	< 500ms	No
widget.deleted	DELETE /widgets/:id	Project	High	widget_id, deleted_at, actor_id, cascade_count	< 200ms	Yes
widget.status_changed	Status transition	Global	High	widget_id, old_status, new_status, reason	< 200ms	Yes
widget.error	Processing failure	Project	Critical	widget_id, error_code, message, stack_trace_id	< 100ms	Yes
system.heartbeat	Timer (30s)	Connection	Low	timestamp, server_id, connection_count	N/A	No

Table 3: IF-WS Event Type Matrix

3.3.3 WebSocket Frame Structure

Figure 2: WebSocket Frame Structure

3.3.4 Protocol Notes

Connection Lifecycle:

a) Client sends HTTP upgrade request with `Sec-WebSocket-Protocol: awp.v2`
b) Server validates API key and returns `101 Switching Protocols`
c) Client subscribes to event channels via `subscribe` message
d) Server delivers events as text frames with JSON payloads
e) Either party may send `ping` / `pong` frames for keep-alive

Connection Limits

Each API key supports a maximum of 100 concurrent WebSocket connections. Exceeding this limit returns HTTP 429.

3.3.5 Requirements

IDD-WS-001 The IF-WS interface shall deliver events within the latency SLA defined in Table 3. Shall

IDD-WS-002 The IF-WS interface shall support automatic reconnection with exponential backoff. Shall

IDD-WS-003 The IF-WS interface should deliver missed events upon reconnection within a 5-minute window. Should

3.4 IF-DB : PostgreSQL Database Interface

3.4.1 Overview

The IF-DB interface defines the internal data persistence layer between the AWP Core Service and the PostgreSQL database cluster. This interface is not exposed to external consumers.

3.4.2 Schema Definitions

Column	Type	Nullable	PK	Default
`#col.name`	`UUID`	No	**PK**	`gen_random_uuid()`
`#col.name`	`VARCHAR(255)`	No	—	`–`
`#col.name`	`widget_status`	No	—	`'active'`
`#col.name`	`JSONB`	Yes	—	`'{}'`
`#col.name`	`INTEGER`	No	—	`1`
`#col.name`	`TIMESTAMPTZ`	No	—	`NOW()`
`#col.name`	`TIMESTAMPTZ`	No	—	`NOW()`

Table 4: IF-DB Schema: `widgets` — Primary widget storage

Column	Type	Nullable	PK	Default
`#col.name`	`BIGSERIAL`	No	**PK**	`auto`
`#col.name`	`UUID`	No	—	`–`
`#col.name`	`VARCHAR(50)`	No	—	`–`
`#col.name`	`JSONB`	Yes	—	`NULL`
`#col.name`	`UUID`	Yes	—	`NULL`
`#col.name`	`TIMESTAMPTZ`	No	—	`NOW()`

Table 5: IF-DB Schema: `widget_events` — Event audit log

3.4.3 SQL Examples

Index creation for query performance:

```sql
CREATE INDEX idx_widgets_status ON widgets (status)
  WHERE status != 'archived';

CREATE INDEX idx_events_widget_id ON widget_events (widget_id)
  INCLUDE (event_type, created_at);
```

3.4.4 Requirements

IDD-DB-001 The IF-DB interface shall enforce referential integrity via foreign key constraints. Shall

IDD-DB-002 The IF-DB interface shall use row-level locking with Shall `SELECT ... FOR UPDATE SKIP LOCKED` for concurrent access.

IDD-DB-003 The IF-DB interface should support point-in-time recovery with a Should retention period of 30 days.

3.5 IF-MQ: RabbitMQ Message Queue Interface

3.5.1 Overview

The IF-MQ interface handles asynchronous message processing between the AWP Core Service and downstream worker services. Messages are routed through a RabbitMQ broker using topic exchanges.

3.5.2 Queue Topology

Figure 3: IF-MQ Queue Topology

Figure 4: IF-MQ Queue Bindings

3.5.3 Message Fields

The following fields are included in all IF-MQ messages. Underlined fields are mandatory in every message. ~~Struck-through~~ fields are deprecated and will be removed in version 3.0.

- message_id — Unique UUID for idempotency
- event_type — Event classification string
- timestamp — ISO 8601 event timestamp
- widget_id — Target widget identifier
- payload — Event-specific data (JSON object)
- actor_id — User or service that triggered the event
- correlation_id — Request tracing identifier
- ~~legacy_type~~ — Old event type format (v1 compat)
- ~~raw_payload~~ — Unstructured payload string (v1 compat)

3.5.4 Message Format Example

```
{
  "message_id": "msg_a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "event_type": "widget.created",
  "timestamp": "2025-01-15T10:30:00.000Z",
  "widget_id": "wgt_7f3a2b1c-9d4e-4a5f-b6c8-1e2d3f4a5b6c",
  "payload": {
    "name": "Weather Dashboard",
    "status": "active",
    "config": { "theme": "dark" }
  },
  "actor_id": "usr_admin_01",
  "correlation_id": "req_trace_xyz789"
}
```

3.5.5 Requirements

IDD-MQ-001 The IF-MQ interface shall guarantee at-least-once delivery for all messages. Shall

IDD-MQ-002 The IF-MQ interface shall support dead-letter queues for messages that fail processing after 3 retry attempts. Shall

IDD-MQ-003 The IF-MQ interface should support message priority levels (1-10) for queue ordering. Should

4 Requirements Traceability

This section maps IDD requirements to source SRS requirements. A total of 13 requirements are defined in this document.

IDD Requirement	Description	SRS Trace
IDD-REST-001	JSON payload conformance	SRS-API-010
IDD-REST-002	Rate limiting enforcement	SRS-API-015
IDD-REST-003	Conditional request support	SRS-API-020
IDD-REST-004	Batch operation support	SRS-API-025
IDD-WS-001	Event delivery latency SLA	SRS-RT-001
IDD-WS-002	Automatic reconnection	SRS-RT-005
IDD-WS-003	Missed event delivery	SRS-RT-010
IDD-DB-001	Referential integrity	SRS-DATA-001
IDD-DB-002	Concurrent access locking	SRS-DATA-005
IDD-DB-003	Point-in-time recovery	SRS-DATA-010
IDD-MQ-001	At-least-once delivery	SRS-MSG-001
IDD-MQ-002	Dead-letter queue support	SRS-MSG-005
IDD-MQ-003	Message priority ordering	SRS-MSG-010

Table 6: Requirements Traceability Matrix

5 Notes

5.1 Glossary

API Application Programming Interface — a set of protocols and tools for building software applications.

AMQP Advanced Message Queuing Protocol — an open standard for message-oriented middleware.

CRUD Create, Read, Update, Delete — the four basic operations of persistent storage.

CSCI Computer Software Configuration Item — a software element treated as a single entity for configuration management purposes.

DID Data Item Description — a form specifying the content and format of a data deliverable.

IDD Interface Design Description — a document describing the interface characteristics of a system or CSCI.

SLA Service Level Agreement — a commitment between a service provider and client defining quality metrics.

WebSocket A communication protocol providing full-duplex channels over a single TCP connection.

5.2 Acronyms

Acronym	Definition
AWP	Acme Widget Platform
CDRL	Contract Data Requirements List
DB	Database
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MQ	Message Queue
REST	Representational State Transfer
RFC	Request for Comments
SRS	Software Requirements Specification
SQL	Structured Query Language
UUID	Universally Unique Identifier
WS	WebSocket
WSS	WebSocket Secure

Table 7: Acronym List

5.3 Assumptions and Clarifications

a) All timestamps use UTC and conform to ISO 8601 format.
b) Widget IDs are globally unique UUID v4 values generated server-side; clients must not generate their own.
c) The **IF-DB** interface assumes PostgreSQL version 15 or later with the `pgcrypto` extension installed.
d) Message queue durability settings assume persistent storage with mirrored queues across at least two broker nodes.
e) API versioning follows URI-based strategy (`/v1/`, `/v2/`); header-based versioning is not supported.

Appendix A: Sample Message Schemas

The following JSON schemas define the structure of **IF-MQ** messages.

Widget Created Event Schema:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": ["message_id", "event_type", "timestamp", "widget_id"],
  "properties": {
    "message_id": { "type": "string", "format": "uuid" },
    "event_type": { "const": "widget.created" },
    "timestamp": { "type": "string", "format": "date-time" },
    "widget_id": { "type": "string", "pattern": "^wgt_" },
    "payload": {
      "type": "object",
      "properties": {
        "name": { "type": "string", "maxLength": 255 },
        "status": { "enum": ["active", "paused", "archived"] }
      }
    }
  }
}
```

Appendix B: Revision History

Version	Date	Description	Author
1.0	2024-06-15	Initial draft — **IF-REST** and **IF-DB** interfaces	J. Smith
1.1	2024-09-01	Added **IF-WS** WebSocket interface	A. Chen
1.5	2024-11-15	Added **IF-MQ** message queue interface	R. Patel
2.0	2025-01-20	Major revision — updated all interfaces for v2 API	Acme Team

Table 8: Document Revision History