AMQP Message Structure for MET-SWIM

Table of Contents

The original working copy of this document is https://github.com/iblsoft/swimdemo/blob/main/MET-SWIM-AMQP-Guidance.md

# 1 Version History

| Version | Date | Changes | Author |
| --- | --- | --- | --- |
| 1.0-draft | 2025-06 | Initial draft specification | EUROCONTROL MET3SG Task Team on Service Architecture |

# 2 Abstract

This guidance document defines the message structure and properties for AMQP 1.0 messages used in the MET-SWIM (Meteorological System Wide Information Management) implementation of the European Union Common Project Regulation (CP1). It provides guidance for developers and organizations implementing meteorological OPMET data publishing systems using AMQP interfaces.

The guidance is developed and maintained by the EUROCONTROL MET3SG Task Team on Service Architecture.

# 3 Introduction

## 3.1 Background

The implementation of MET-SWIM under EU Common Project Regulation (CP1) requires standardized interfaces for the exchange of meteorological data. While EUROCONTROL has published service definitions for SWIM services, there is currently a lack of detailed standardization regarding the specific AMQP message properties and application properties to be used.

This specification addresses this gap by defining a standard approach for AMQP message structure, ensuring interoperability between different organizations and systems implementing MET-SWIM services.

This specification is inspired by the WMO WIS2 Notification Message standard while being tailored to the specific requirements of MET-SWIM and the European aviation meteorological community.

## 3.2 Objectives

* Standardize AMQP message properties for meteorological data exchange
* Enable efficient filtering of messages
* Ensure interoperability between different MET-SWIM implementations
* Provide forward compatibility for future meteorological data types beyond CP1

## 3.3 Authors

This specification is developed and maintained by the **EUROCONTROL MET3SG Task Team on Service Architecture**.

Contributions to this specification:

* **Michal Weis** ([Michal.Weis@iblsoft.com](mailto:Michal.Weis@iblsoft.com))
  + The original proposal of adapting WIS 2 WNM notification metadata to AMQP application properties.
* **Boris Burger** ([Boris.Burger@iblsoft.com](mailto:Boris.Burger@iblsoft.com))
  + Alignment with AMQP and WNM specifications.
  + Relationship of message and application properties to IWXXM data model.
* **Florian Dobener** ([Florian.Dobener@dwd.de](mailto:Florian.Dobener@dwd.de))
  + Structure of technical messages pertaining to the queues.

# 4 Scope

## 4.1 Current Scope (CP1)

This specification currently covers the following meteorological report types as defined in ICAO Annex 3:

* METAR
* SPECI
* TAF
* SIGMET

## 4.2 Future Scope

The specification is expected to accommodate future additions to data distributed using MET-SWIM publishing and subscription sservices including:

* Radar data
* Automated Weather Observing System (AWOS) data
* Model Output Statistics (MOS)
* Other value-added meteorological products

# 5 References

* [OASIS AMQP 1.0 Specification, Part 3: Messaging](https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html)
* [AMQP Filter Expressions 1.0](https://docs.oasis-open.org/amqp/filtex/v1.0/filtex-v1.0.html)
* [EU Common Project Regulation (CP1)](https://eur-lex.europa.eu/eli/reg_impl/2021/116/oj/eng)
* [WMO WIS 2.0 Overview](https://community.wmo.int/en/activity-areas/wis/WIS2-overview)
* [WMO WIS 2.0 Notification Message Standard](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html)
* [RFC 3339 - Date and Time on the Internet](https://www.rfc-editor.org/rfc/rfc3339.html)
* [ICAO Annex 3 - Meteorological Service for International Air Navigation](https://store.icao.int/en/annex-3-meteorological-service-for-international-air-navigation)
* [OGC API - Environmental Data Retrieval (EDR)](https://ogcapi.ogc.org/edr/)
* [IWXXM METAR-SPECI Subscription and Request Service 1.0](https://eur-registry.swim.aero/services/eurocontrol-iwxxm-metar-speci-subscription-and-request-service-10)
* [IWXXM SIGMET Subscription and Request Service 1.0](https://eur-registry.swim.aero/services/eurocontrol-iwxxm-sigmet-subscription-and-request-service-10)
* [IWXXM TAF Subscription and Request Service 1.0](https://eur-registry.swim.aero/services/eurocontrol-iwxxm-taf-subscription-and-request-service-10)

# 6 AMQP Message Structure Overview

An AMQP 1.0 message in MET-SWIM consists of the following components:

1. **Message Addressing** - Destination address using a hierarchical structure
2. **Message Header** - Priority and TTL settings
3. **Message Properties** - Standard AMQP properties including subject, content-type, etc.
4. **Application Properties** - Custom properties for meteorological data identification and filtering
5. **Message Payload** - The actual data (IWXXM XML).

# 7 Message Addressing

## 7.1 Address Structure

MET-SWIM uses a simplified three-level hierarchical structure for message addresses, inspired by the [WIS 2 Topic Hierarchy](https://wmo-im.github.io/wis2-topic-hierarchy/standard/wis2-topic-hierarchy-STABLE.html).

For meteorological data, the address structure is:

weather.aviation.<report-type>

Where <report-type> is one of:

* metar - for METAR and SPECI reports
* taf - for TAF reports
* sigmet - for SIGMET reports

Technical messages are delivered to the same addresses as the meteorological data they pertain to, but are distinguished by their message properties (see Section 11).

## 7.2 Relationship to WIS 2 Topic Hierarchy

The WIS 2 Topic Hierarchy defines a comprehensive structure for global weather data exchange:

origin/a/wis2/{centre-id}/data/{data-policy}/{earth-system-discipline}/{...}

MET-SWIM simplifies this structure for the specific needs of aviation meteorological data exchange:

1. **Simplified Structure**: Instead of the full WIS 2 path, we use a concise three-level structure focusing on the domain (weather), sub-domain (aviation), and data type.
2. **Aviation Focus**: The WIS 2 weather/aviation hierarchy currently defines only metar, taf, and qvaci (quantitative volcanic ash concentration). MET-SWIM extends this to include sigmet which, while not yet in the WIS 2 aviation hierarchy, is essential for aviation safety.
3. **Centre Identification**: Unlike WIS 2, which embeds the issuing centre ID in the topic, MET-SWIM does not currently indicate the issuing centre in the address (“topic”).

**TODO:** Do we need any indication of the original organisation who was issuing the data?

The IWXXM data model does not indicate who was the originator of the IWXXM report. It does indicate the affected aerodrome oir airspace, but the issuing organisation is not known in the data model. Only for IWXXM reports translated from TAC the name and designator of the translation centre is included.

## 7.3 Examples

weather.aviation.metar  
weather.aviation.taf  
weather.aviation.sigmet

## 7.4 Wildcard Addresses in Subscriptions

Consumers can subscribe using wildcards to receive messages from multiple related addresses only if the AMQP broker supports wildcard functionality. **Wildcard support is not part of the AMQP 1.0 specification** - it is implemented as a broker-specific extension. Wildcard support is common in the popular brokers, but is not guaranteed.

### 7.4.1 Broker Extension: Address Wildcards

Many brokers implement wildcard extensions that allow pattern matching in subscription addresses:

**Common Wildcard Characters:**

* \* - matches exactly one level in the hierarchy
* # - matches zero or more levels from that point onwards

**Wildcard Examples:**

weather.aviation.\* # Subscribe to all aviation weather types  
weather.# # Subscribe to all weather data  
#.metar # Subscribe to METAR and SPECI

### 7.4.2 Broker-Specific Wildcard Implementation

Different AMQP 1.0 brokers implement wildcard support with varying levels of compatibility:

#### 7.4.2.1 Apache ActiveMQ Artemis

* Full wildcard support using \* and # characters
* Wildcards work with both queues and topics

#### 7.4.2.2 Apache Qpid Proton

* Supports both \* and # wildcards

#### 7.4.2.3 RabbitMQ (with AMQP 1.0 plugin)

* No direct wildcard support for \* and # in AMQP 1.0 addressing
* RabbitMQ’s AMQP 1.0 plugin maps addresses to internal AMQP 0.9.1 exchanges/queues
* Wildcards only work if addresses are explicitly mapped to topic exchanges with routing patterns

**TODO**: Florian Dobener (DWD) to verify this assessment, as DWD uses RabbitMQ in their implementation. These webpages seem to suggest that RabbitMQ does indeed support the \* and # wildcards:

* <https://www.cloudamqp.com/blog/rabbitmq-topic-exchange-explained.html>
* <https://www.rabbitmq.com/tutorials/tutorial-five-spring-amqp>

# 8 AMQP Header Properties

The subsequent section define the AMQP 1.0 message properties used in MET-SWIM implementations. For detailed technical specifications, refer to the [OASIS AMQP 1.0 Specification, Part 3: Messaging](https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html).

The normative documentation for the AMQP headers is in AMQP 1.0 specification, Part 3: Messaging - [Section 3.2.5 - Header](https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#type-header).

## 8.1 Priority

**TODO**: MET3SG Task Team on Service Architecture - review and confirm priority values for different message types.

1. Do we need to prescribe exact priority numbers or should this be left to the implementers?
2. If not, should we at least suggest, that SIGMET and SPECI should have higher priority than other OPMET report types, but without prescribing the exact numeric value?

The priority field (0-9, where 9 is highest) SHALL be set according to the operational importance:

| Report Type | Priority | Rationale |
| --- | --- | --- |
| SIGMET | 7 | Highest priority for safety-critical information |
| SPECI | 7 | Special reports indicating significant changes |
| TAF | 5 | Forecast information with medium priority |
| METAR | 4 | Routine observations |
| Other data | 3 | Default for unspecified data types |
| GRIB | 1-2 | Large volume data with lower priority |

# 9 AMQP Standard Message Properties

See AMQP 1.0 specification, Part 3: Messaging, [Section 3.2.6 - Properties](https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#type-properties) for the normative documentation.

## 9.1 subject (MANDATORY)

**Note**: While the subject property is optional in the [AMQP 1.0 specification](https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#type-properties), it is **MANDATORY** in MET-SWIM to enable effective message filtering and processing.

The subject property provides a summary of the message content and purpose, allowing consumers to quickly identify:

1. **Message Category**: Whether this is a meteorological report, technical message, or vendor-specific extension
2. **Message Content**: For meteorological data, the specific report details
3. **Message Purpose**: Normal operations, amendments, corrections, or administrative functions

**For Meteorological Reports**, the subject SHALL follow the pattern:

DATA\_<REPORT-TYPE>\_<LOCATION>\_<STATUS>\_<ISSUE-TIME>

Where:

* <REPORT-TYPE>: TAF, METAR, SPECI, or SIGMET
* <LOCATION>: 4-letter ICAO location identifier of an aerodrome or an airspace
* <STATUS>:
  + NORMAL - signifies intitial issue of a report
  + AMENDMENT - signifies a report amendment
  + CORRECTION - signifies a report correction
* <ISSUE-TIME>: UTC issue time in format YYYYMMDDHHmmss

Examples:

DATA\_TAF\_LOWS\_NORMAL\_20250401051500  
DATA\_TAF\_LOWS\_AMENDMENT\_20250401064500  
DATA\_METAR\_LOWS\_CORRECTION\_20250401064500  
DATA\_SIGMET\_UDDD\_NORMAL\_20250415141000

**For Technical Messages**, the subject SHALL follow structured patterns based on message type:

* **Subscription Status and Maintenance Messages**: TECHNICAL\_<MESSAGE-TYPE>\_<QUEUE-STATUS>\_<TIMESTAMP>
* **Error Messages**: TECHNICAL\_ERROR\_<TIMESTAMP>

Examples:

TECHNICAL\_SUBSCRIPTION-STATUS\_PAUSED\_20250415143000  
TECHNICAL\_MAINTENANCE\_PAUSED\_20250415120000  
TECHNICAL\_MAINTENANCE\_ACTIVE\_20250415150000  
TECHNICAL\_ERROR\_20250415143215

See [Section 11 - Technical Messages](#technical-messages) for complete details on technical message structure.

**For Vendor-Specific Extensions**, implementers MAY define custom subject patterns using the EXTENSION\_ prefix, but such patterns SHOULD clearly distinguish extension messages from standard data and technical messages.

## 9.2 content-type (MANDATORY)

Indicates the MIME type of the payload:

**For Meteorological Data:**

* application/xml - for IWXXM XML data
* application/uri-list - for messages containing only URLs to larger datasets

**TODO:** Check if the application/uri-list for the payload is actually useful if we are putting the links into application metadata. This could be just an unnecesary duplicity.

**For Technical Messages:**

* application/json - for technical/administrative messages (see [Section 11 - Technical Messages](#technical-messages))

**For Vendor-Specific Extensions:**

* Other MIME types MAY be used for vendor-specific message formats outside the scope of this specification

## 9.3 content-encoding (CONDITIONAL)

Required when compression is applied:

* gzip - for compressed data
* identity - for uncompressed data (default if omitted)

## 9.4 absolute-expiry-time (OPTIONAL)

**TODO:** Does is make sense to prescribe how long the messages should persist in the durable queues by the report type? Or should this be just a suggestion and left for implementations to decide how long METAR/TAF/SIGMET stay in the durable queues before they expire?

Unix timestamp indicating when the message expires in durable queues:

* METAR/SPECI: Current time + 3 hours
* TAF: Current time + 12 hours
* SIGMET: Current time + 24 hours

# 10 AMQP Application Properties

Application properties enable server-side filtering using [AMQP Filter Expressions 1.0](https://docs.oasis-open.org/amqp/filtex/v1.0/filtex-v1.0.html). Properties are divided into mandatory, conditional, and optional categories based on their usage requirements.

[Section 3.2.7 - Application Properties](https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#type-application-properties)

## 10.1 Mandatory Application Properties

These properties MUST be present in all meteorological data messages.

### 10.1.1 conformsTo (conditional)

**TODO**: Should the link to the service definition in the SWIM registry be mandatory? If a producer has some data they want to distribute via MET-SWIM AMQP, must they always register it first? Or should we make this property optional?

Identifies the specification version and the corresponding SWIM service definition:

conformsTo: "https://eur-registry.swim.aero/services/eurocontrol-iwxxm-[service-type]-10"

Example values:

* https://eur-registry.swim.aero/services/eurocontrol-iwxxm-metar-speci-subscription-and-request-service-10
* https://eur-registry.swim.aero/services/eurocontrol-iwxxm-taf-subscription-and-request-service-10
* https://eur-registry.swim.aero/services/eurocontrol-iwxxm-sigmet-subscription-and-request-service-10

### 10.1.2 topic (mandatory)

The actual address/topic used for the message:

topic: "weather.aviation.metar"

### 10.1.3 properties.icao\_location\_identifier (conditional)

ICAO identifier of the location. Mandatory for reports that are issued for an aerodrome or an airspace that can be identified by a 4-letter ICAO location designator.

properties.icao\_location\_identifier: "EBBR"

### 10.1.4 properties.icao\_location\_type (conditional)

Type of location. Mandatory for reports that are issued for an aerodrome or an airspace that can be identified by a 4-letter ICAO location designator.

* AD - Aerodrome (for METAR/TAF)
  + From AIXM [AirportHeliportType](https://aixm.aero/sites/default/files/imce/AIXM51HTML/AIXM/DataType_CodeAirportHeliportType.html)
* FIR - Flight Information Region (for SIGMET)
  + From AIXM [AirspaceType](https://aixm.aero/sites/default/files/imce/AIXM511HTML/AIXM/DataType_CodeAirspaceType.html)
* UIR - Upper Information Region
  + From AIXM [AirspaceType](https://aixm.aero/sites/default/files/imce/AIXM511HTML/AIXM/DataType_CodeAirspaceType.html)
* CTA - Control Area
  + From AIXM [AirspaceType](https://aixm.aero/sites/default/files/imce/AIXM511HTML/AIXM/DataType_CodeAirspaceType.html)
* OTHER:FIR\_UIR - Combined FIR/UIR
  + This is the FIR/UIR encoding used in aixm:type in <https://schemas.wmo.int/iwxxm/2025-2RC1/examples/sigmet-A6-1a-TS.xml>

## 10.2 Temporal Properties

These properties contain date and time information extracted from the IWXXM document.

### 10.2.1 properties.pubtime (mandatory)

RFC 3339 formatted publication/issue time extracted from iwxxm:issueTime:

properties.pubtime: "2025-04-15T14:10:00Z"

This property is mandatory for all message types.

## 10.3 Conditional Properties - Temporal

These properties are required based on the message type:

### 10.3.1 properties.datetime (conditional)

For METAR/SPECI only - observation time in RFC 3339 format from iwxxm:observationTime:

properties.datetime: "2025-03-31T03:00:00Z"

### 10.3.2 properties.start\_datetime (conditional)

For TAF/SIGMET - start of validity period in RFC 3339 format from iwxxm:validPeriod:

properties.start\_datetime: "2025-04-15T14:30:00Z"

### 10.3.3 properties.end\_datetime (conditional)

For TAF/SIGMET - end of validity period in RFC 3339 format:

properties.end\_datetime: "2025-04-15T18:00:00Z"

## 10.4 Conditional Properties - External Links

Link properties are required when the message payload does not contain the IWXXM data. For embedded IWXXM data (typical for METAR, SPECI, TAF, SIGMET), links are optional.

For meteorological data messages, when the report content is not embedded in the AMQP payload, these properties SHALL be used by data producers to provide URL links for download.

When there is no payload in the meteorological data message, at least one download link SHALL be provided.

### 10.4.1 links.count (conditional)

Number of links provided in the message. This property indicates how many link entries are included in the message, allowing consumers to know the expected number of links without probing for them.

This property is **optional** if the meteorological data is stored in the payload. The property is **mandatory** when payload does not contain meteorological data, in this case the download URL needs to be provided as a link.

Example:

links.count: 1

### 10.4.2 links[0].href (conditional)

Primary URL for data retrieval:

links[0].href: "https://swim.example.com/filedb/DATA\_TAF\_LOWI\_NORMAL\_20250414111500.xml"

### 10.4.3 links[0].rel (conditional)

Link relationship type, following standards defined in [IANA Link Relation Types](https://www.iana.org/assignments/link-relations/link-relations.xml) and [WMO WIS Link Types](https://codes.wmo.int/wis/link-type), aligned with the [WIS 2.0 Notification Message Standard](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html).

The [WMO WIS Link Types registry](https://codes.wmo.int/wis/link-type) defines meteorological-specific link relations that extend beyond the standard IANA set, specifically designed for weather, climate, and water data exchange scenarios.

**Standard Link Relations:**

| Value | Usage | Description | Source Standard |
| --- | --- | --- | --- |
| canonical | Primary data link | Points to the preferred/authoritative version of the resource | [IANA Link Relations](https://www.iana.org/assignments/link-relations/link-relations.xml) |
| item | Alternative access | Links to alternative representations or access methods (e.g., EDR API, JSON format) | [IANA Link Relations](https://www.iana.org/assignments/link-relations/link-relations.xml) |
| update | Amendments/Corrections | For amended or corrected reports (TAF AMD, METAR COR) | [WMO WIS Link Types](https://codes.wmo.int/wis/link-type) |

**Implementation Notes:**

* **canonical**: MUST be used for the primary/original IWXXM data link
* **item**: SHOULD be used for EDR API links, alternative formats, or related collections
* **update**: SHOULD be used for links to amended or corrected meteorological reports (following WIS 2.0 WNM standard)

**Examples by Report Status:**

# Original report  
links[0].rel: "canonical"  
  
# Amended report   
links[0].rel: "update"  
  
# Additional EDR access  
links[1].rel: "item"

### 10.4.4 links[0].type (conditional)

Content type (MIME type) of the linked resource, following [IANA Media Types](https://www.iana.org/assignments/media-types/media-types.xhtml) registry and aligned with [WIS 2.0 WNM standard](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html) practices.

**Standard MIME Types for MET-SWIM:**

| Type | Usage | Description | Examples |
| --- | --- | --- | --- |
| application/xml | IWXXM data | XML-encoded meteorological reports | METAR, TAF, SIGMET in IWXXM format |
| application/zip | Compressed collections | Multiple reports or large datasets | EDR API responses with multiple reports |
| application/json | JSON representations | Alternative JSON format of meteorological data | EDR API JSON output |
| application/netcdf | Gridded data | Network Common Data Form - widely used for climate and weather model data | NWP model output, satellite data, climate datasets |
| application/grib | Gridded data | Numerical weather prediction data in GRIB format | GRIB2 model output |
| application/bufr | Binary data | BUFR-encoded observations | Traditional BUFR observations |

**Implementation Notes:**

* **IWXXM XML data**: MUST use application/xml
* **Compressed data**: SHOULD use application/zip for multi-file collections
* **EDR API responses**: MAY use application/json or application/zip depending on response format
* **Custom types**: MUST follow IANA Media Types registry or use application/octet-stream for unspecified binary data

**Examples:**

# Primary IWXXM link  
links[0].type: "application/xml"  
  
# EDR API collection response  
links[1].type: "application/zip"  
  
# JSON alternative representation  
links[2].type: "application/json"

## 10.5 Optional Additional Links (when payload is embedded)

Even when IWXXM data is embedded in the payload, additional links MAY be provided for alternative access methods:

* **EDR API links**: Links to Environmental Data Retrieval (EDR) services that provide the same data in different formats or through query interfaces
* **Alternative format links**: Links to the same data in different formats (e.g., JSON representations)
* **Collection links**: Links to related data collections or time series

Example of optional additional links:

links.count: 2  
links[0].href: "https://swim.example.com/filedb/DATA\_TAF\_LPFL\_NORMAL\_20250415140000.xml"  
links[0].rel: "canonical"  
links[0].type: "application/xml"  
links[1].href: "https://edr.swim.example.com/edr/collections/iwxxm-taf/locations/icao:LPFL?datetime=2025-04-15T15:00:00Z/2025-04-16T00:00:00Z"  
links[1].rel: "item"  
links[1].type: "application/zip"

## 10.6 Optional Properties - Payload Integrity

Integrity properties provide cryptographic verification of message payload content, enabling detection of data corruption during transmission and ensuring data authenticity. These properties are aligned with the [WIS 2.0 Notification Message Standard](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html) integrity mechanism.

### 10.6.1 properties.integrity.method (optional)

Specifies the cryptographic hash algorithm used for payload verification. The hashes are the ones preferred in WIS 2 WNM, with sha512 as the main recommendation.

| Method | Description | Implementation Requirement | Programming Language Support |
| --- | --- | --- | --- |
| sha512 | SHA-512 hash algorithm | **MUST support** | Universal (Python, Java, C++, JavaScript, .NET) |
| sha256 | SHA-256 hash algorithm | **MUST support** | Universal (Python, Java, C++, JavaScript, .NET) |
| sha384 | SHA-384 hash algorithm | **SHOULD support** | Universal (Python, Java, C++, JavaScript, .NET) |
| sha3-256 | SHA-3 256-bit hash | **MAY support** | Python 3.6+, Java 9+, OpenSSL 1.1.1+, Node.js |
| sha3-384 | SHA-3 384-bit hash | **MAY support** | Python 3.6+, Java 9+, OpenSSL 1.1.1+, Node.js |
| sha3-512 | SHA-3 512-bit hash | **MAY support** | Python 3.6+, Java 9+, OpenSSL 1.1.1+, Node.js |

**Implementation Requirements:**

* **Producers SHOULD use sha512** as the preferred method for new implementations (widely supported, strong security)
* **Consumers MUST support** at least sha256 and sha512 methods for maximum interoperability
* **Consumers SHOULD support** sha384 for complete SHA-2 family compatibility
* **Consumers MAY support** SHA-3 variants (sha3-256, sha3-384, sha3-512) for future compatibility.
* The method field is case-sensitive and MUST be lowercase

### 10.6.2 properties.integrity.value (optional)

Contains the Base64-encoded hash value of the message payload, following the [WIS 2.0 WNM standard](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html) encoding format:

properties.integrity.method: "sha512"  
properties.integrity.value: "gmUfpr2K48cc5SjVm/Eo6vT/BDX6LGRyPIkG9jk3...P3SqfSCw=="

**Hash Calculation:**

1. **For compressed payloads**: Calculate hash on the **uncompressed** data (original IWXXM XML)
2. **For uncompressed payloads**: Calculate hash directly on the payload bytes
3. **Encoding**: Present hash as **Base64-encoded string** (aligned with WIS 2.0 WNM standard)
4. **Character encoding**: Use UTF-8 for text data before hashing

## 10.7 Geometry Properties

The optional geometry properties enable spatial filtering of meteorological messages and provide location context. These properties are inspired by the [WMO WIS 2.0 Notification Message Standard](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html) which uses GeoJSON (RFC7946) geometry objects.

However, MET-SWIM uses a simplified geometry representation optimized for AMQP server-side filtering using SQL-like expressions, rather than full GeoJSON compliance.

### 10.7.1 geometry.type (optional)

Specifies the type of geometric representation:

| Value | Usage | Description |
| --- | --- | --- |
| Point | METAR, SPECI, TAF | Represents aerodrome coordinates as a single point |
| Bounding-box | SIGMET | Represents area coverage using a rectangular bounding box |

**Relationship to Standards:**

* **GeoJSON Standard**: Uses Point and Polygon geometry types
* **WIS 2.0**: Follows GeoJSON with Point for stations and Polygon for areas
* **MET-SWIM**: Uses simplified Bounding-box instead of Polygon for easier SQL-like filtering in AMQP brokers

The Bounding-box type is a MET-SWIM extension that simplifies spatial queries compared to full polygon representations. While GeoJSON and WIS 2.0 would represent areas using Polygon geometry with multiple coordinate pairs, Bounding-box uses only four coordinate values that can be easily filtered using SQL expressions like:

geometry.type = 'Bounding-box'   
 AND geometry.coordinates.latitude-min <= 45.0  
 AND geometry.coordinates.latitude-max >= 44.0

### 10.7.2 Point Geometry Properties (optional)

For point locations (aerodromes in METAR, SPECI, TAF):

geometry.type: "Point"  
geometry.coordinates.latitude: 48.1628  
geometry.coordinates.longitude: 17.1785

**Properties:**

* **geometry.coordinates.latitude**: Decimal degrees latitude (WGS84) of the aerodrome reference point
* **geometry.coordinates.longitude**: Decimal degrees longitude (WGS84) of the aerodrome reference point

**Value Ranges:**

* Latitude: -90.0 to +90.0 (negative for South, positive for North)
* Longitude: -180.0 to +180.0 (negative for West, positive for East)

### 10.7.3 Bounding Box Geometry Properties (optional)

For area-based data (SIGMET covering FIR/UIR regions):

geometry.type: "Bounding-box"  
geometry.coordinates.latitude-min: 38.83  
geometry.coordinates.latitude-max: 41.3  
geometry.coordinates.longitude-min: 43.47  
geometry.coordinates.longitude-max: 46.63

**Properties:**

* **geometry.coordinates.latitude-min**: Minimum (southernmost) latitude of the bounding rectangle
* **geometry.coordinates.latitude-max**: Maximum (northernmost) latitude of the bounding rectangle
* **geometry.coordinates.longitude-min**: Minimum (westernmost) longitude of the bounding rectangle
* **geometry.coordinates.longitude-max**: Maximum (easternmost) longitude of the bounding rectangle

**Coordinate System**: World Geodetic System 1984 (WGS84)

**Value Ranges**: Same as Point geometry (latitude: ±90°, longitude: ±180°)

**Validation Rules:**

* latitude-min ≤ latitude-max
* longitude-min ≤ longitude-max (except when crossing the antimeridian)
* All coordinate values must be valid WGS84 coordinates

### 10.7.4 Spatial Filtering Examples

The geometry properties enable server-side filtering using AMQP Filter Expressions:

-- Filter by specific aerodrome coordinates (within 200km radius conceptually)  
geometry.type = 'Point'   
 AND geometry.coordinates.latitude BETWEEN 47.0 AND 49.0  
 AND geometry.coordinates.longitude BETWEEN 16.0 AND 18.0  
  
-- Filter SIGMET affecting specific geographic region  
geometry.type = 'Bounding-box'  
 AND geometry.coordinates.latitude-max >= 45.0  
 AND geometry.coordinates.latitude-min <= 50.0  
 AND geometry.coordinates.longitude-min >= 15.0  
 AND geometry.coordinates.longitude-max <= 20.0  
  
-- Combined filtering with location type  
geometry.type = 'Point'   
 AND properties.icao\_location\_type = 'AD'  
 AND geometry.coordinates.latitude BETWEEN 45 AND 50  
 AND geometry.coordinates.longitude BETWEEN 15.6 AND 20.7

### 10.7.5 Data Source for Coordinates

Coordinate values SHALL be extracted from IWXXM XML documents:

* **Aerodrome coordinates**: From iwxxm:aerodrome//aixm:ARP/aixm:ElevatedPoint/gml:pos in aerodrome references
* **SIGMET area coordinates**: Calculated as bounding box of the union of the horizontal position of the hazards represented in the various iwxxm:geometry elements. Note that it is not trivial to calculate bounding boxes of e.g. tropical cyclone SIGMETs specified using a set of circles with a radius.

See [Appendix B: Property Extraction from IWXXM](#Xf6c8d0e4dae9eb627b5cb696ecd1a18ad476c2a) for detailed extraction procedures.

# 11 Message Payload

## 11.1 Small Data (Embedded)

For IWXXM XML documents, the payload SHALL contain the complete IWXXM XML document

* The message property content-type must be set to application/xml.
* The payload can be optionally compressed using gzip, in this case content-encoding message property SHALL be set to gzip.

## 11.2 Large Data (Referenced)

TODO: Michal Weis where did the idea of the application/uri-list come from? If we are storing links in application properties, do we need them in the payload?

For larger datasets where embedding the data in the payload is not practical:

* Payload contains URL(s) for retrieval
* content-type: application/uri-list
* Link properties in application properties are MANDATORY
* links.count property indicates the number of available links

For smaller datasets (typical METAR, SPECI, TAF, SIGMET):

* IWXXM data is embedded in the payload with content-type: application/xml
* Link properties are OPTIONAL and may provide alternative access methods (EDR services, alternative formats, etc.)

# 12 Technical Messages

## 12.1 Overview

Technical messages are administrative messages that provide information about the status of specific topics (addresses) or queues. These messages are separate from meteorological data messages and are used to inform consumers about:

* Queue or subscription status changes
* Error conditions
* Planned maintenance windows

Technical messages are delivered to the same addresses as regular meteorological messages but are distinguished by their message properties, specifically the subject field.

## 12.2 Message Structure

### 12.2.1 Message Properties (Technical Messages)

Technical messages use the standard AMQP message properties with the following specifications:

* **subject**: Follows a structured pattern based on message type (see patterns below)
* **content-type**: application/json
* **content-encoding**: identity (no compression required for small JSON payloads)

**Subject Patterns for Technical Messages:**

For subscription status and maintenance messages:

TECHNICAL\_<MESSAGE-TYPE>\_<QUEUE-STATUS>\_<TIMESTAMP>

For error messages:

TECHNICAL\_ERROR\_<TIMESTAMP>

Where:

* <MESSAGE-TYPE>: SUBSCRIPTION-STATUS or MAINTENANCE
* <QUEUE-STATUS>: PAUSED, ACTIVE, CREATED, DELETED, or INTERRUPTED
* <TIMESTAMP>: Message creation time in format YYYYMMDDHHmmss

Examples:

TECHNICAL\_SUBSCRIPTION-STATUS\_PAUSED\_20250415143000  
TECHNICAL\_MAINTENANCE\_PAUSED\_20250415120000  
TECHNICAL\_MAINTENANCE\_ACTIVE\_20250415150000  
TECHNICAL\_ERROR\_20250415143215

### 12.2.2 Application Properties (Technical Messages)

Technical messages SHALL include the following application properties:

* **conformsTo**: https://eur-registry.swim.aero/services/technical-message-service-10
* **topic**: The address to which this message is sent (e.g., weather.aviation.metar)
* **properties.message\_type**: technical-message

## 12.3 Payload Format

The payload of technical messages SHALL be a JSON object with the following structure:

{  
 "id": "<uuid>",  
 "type": "<subscription-status-message|maintenance-message|error-message>",  
 "queue-status": "<interrupted|paused|active|created|deleted|null>",  
 "timestamp": "<RFC3339 UTC timestamp>",  
 "start": "<RFC3339 UTC timestamp or null>",  
 "duration": "<milliseconds or null>",  
 "message": "<descriptive text>"  
}

## 12.4 Message Types

### 12.4.1 Subscription Status Message

Indicates a status change of the subscription or queue/channel.

| Field | Value | Description |
| --- | --- | --- |
| id | UUID | Unique message identifier |
| type | subscription-status-message | Fixed value |
| queue-status | interrupted, paused, active, created, or deleted | New status of the queue |
| timestamp | RFC3339 UTC | Message creation time |
| start | null | Not used for status messages |
| duration | null | Not used for status messages |
| message | String | Descriptive text about the status change |

Example:

{  
 "id": "550e8400-e29b-41d4-a716-446655440000",  
 "type": "subscription-status-message",  
 "queue-status": "paused",  
 "timestamp": "2025-04-15T14:30:00Z",  
 "start": null,  
 "duration": null,  
 "message": "Queue paused for maintenance"  
}

### 12.4.2 Error Message

Reports error conditions such as validation errors or missing data.

| Field | Value | Description |
| --- | --- | --- |
| id | UUID | Unique message identifier |
| type | error-message | Fixed value |
| queue-status | null | Not applicable for error messages |
| timestamp | RFC3339 UTC | Message creation time |
| start | null | Not used for error messages |
| duration | null | Not used for error messages |
| message | String | Error description |

Example:

{  
 "id": "660e8400-e29b-41d4-a716-446655440001",  
 "type": "error-message",  
 "queue-status": null,  
 "timestamp": "2025-04-15T14:32:15Z",  
 "start": null,  
 "duration": null,  
 "message": "IWXXM validation error: Missing required element iwxxm:issueTime"  
}

### 12.4.3 Maintenance Message

Announces planned maintenance windows or maintenance completion.

| Field | Value | Description |
| --- | --- | --- |
| id | UUID | Unique message identifier |
| type | maintenance-message | Fixed value |
| queue-status | active or paused | Current or expected queue status |
| timestamp | RFC3339 UTC | Message creation time |
| start | RFC3339 UTC or null | Start time of maintenance (null if maintenance end message) |
| duration | Milliseconds or null | Expected duration (null if maintenance end message) |
| message | String | Maintenance description |

Example (maintenance start):

{  
 "id": "770e8400-e29b-41d4-a716-446655440002",  
 "type": "maintenance-message",  
 "queue-status": "paused",  
 "timestamp": "2025-04-15T12:00:00Z",  
 "start": "2025-04-15T14:00:00Z",  
 "duration": 3600000,  
 "message": "Scheduled maintenance window - service will be unavailable"  
}

Example (maintenance end):

{  
 "id": "880e8400-e29b-41d4-a716-446655440003",  
 "type": "maintenance-message",  
 "queue-status": "active",  
 "timestamp": "2025-04-15T15:00:00Z",  
 "start": null,  
 "duration": null,  
 "message": "Maintenance completed - service resumed"  
}

## 12.5 Implementation Notes

1. **Delivery**: Technical messages are delivered to the same addresses as the meteorological data they pertain to. This ensures that all consumers of a specific data type automatically receive relevant technical messages.
2. **Priority**: Technical messages SHOULD use priority level 8 or 9 to ensure timely delivery.
3. **Retention**: Technical messages MAY have shorter retention times than meteorological data:
   * Error messages: 1 hour
   * Status messages: 3 hours
   * Maintenance messages: 24 hours
4. **Filtering**: Consumers MUST distinguish technical messages from meteorological data using:

* -- Filter all meteorological data  
  subject LIKE 'DATA\_%'  
    
  -- Filter all technical messages  
  subject LIKE 'TECHNICAL\_%'  
    
  -- Filter by specific meteorological report type  
  subject LIKE 'DATA\_METAR\_%'  
  subject LIKE 'DATA\_TAF\_%'  
  subject LIKE 'DATA\_SIGMET\_%'  
    
  -- Filter by specific technical message type  
  subject LIKE 'TECHNICAL\_MAINTENANCE\_%'  
  subject LIKE 'TECHNICAL\_SUBSCRIPTION-STATUS\_%'  
  subject LIKE 'TECHNICAL\_ERROR\_%'  
    
  -- Filter by queue status in technical messages  
  subject LIKE 'TECHNICAL\_%\_PAUSED\_%'  
  subject LIKE 'TECHNICAL\_%\_ACTIVE\_%'  
    
  -- Filter using JSON payload content (for additional criteria)  
  subject LIKE 'TECHNICAL\_%' AND JSON\_VALUE(body, '$.type') = 'maintenance-message'

1. **Processing**: Consumer applications SHOULD:
   * Check the subject field to distinguish technical messages from meteorological data
   * Parse the subject pattern to extract message type, status, and timestamp
   * Handle technical messages separately from meteorological data
   * Parse the JSON payload to get detailed information about the technical message
   * Update their internal status based on queue-status changes
   * Alert operators about error conditions
   * Schedule around maintenance windows

# 13 Broker and Client Library Compatibility

## 13.1 AMQP 1.0 Brokers Successfully Used in MET-SWIM Implementations

The following AMQP 1.0 brokers have been successfully used in MET-SWIM implementations:

| Broker | Wildcard Support | Filter Expressions | Notes |
| --- | --- | --- | --- |
| Apache ActiveMQ Artemis | ✓ | ✓ | Full support for SQL-like filtering |
| RabbitMQ | ✓ | ✓ | Requires AMQP 1.0 plugin enabled |
| Apache Qpid Broker-J | ✓ | ✓ | Apache’s AMQP 1.0 broker implementation |

## 13.2 AMQP 1.0 Client Libraries Known to Work in MET-SWIM Context

The following client libraries have been successfully used with AMQP 1.0 brokers in real implementations:

**TODO:** This section needs to be expanded based on experience of other organisations. IBL knows that Qpid Proton C++ and Python APIs work well. In Python specifically, we have so far not been successful with other Python modules besides qpid-proton.

### 13.2.1 JavaScript/Node.js

* **Rhea** - A fully compliant AMQP 1.0 library that is actively maintained. This is the current recommended JavaScript library for AMQP 1.0 connections. Available at <https://github.com/amqp/rhea>.

### 13.2.2 Python

* **Qpid Proton Python API** - Part of [Apache Qpid Proton](https://qpid.apache.org/proton/index.html), this is a mature and well-tested library with full AMQP 1.0 support. Successfully used with ActiveMQ Artemis, RabbitMQ, and other AMQP 1.0 brokers.

### 13.2.3 C++

* **Qpid Proton C++** - Part of [Apache Qpid Proton](https://qpid.apache.org/proton/index.html), verified to work properly with AMQP 1.0 brokers. Provides a portable C implementation with C++ bindings.

### 13.2.4 Java

* **Apache Qpid Proton-J** - Java implementation from [Apache Qpid Proton](https://qpid.apache.org/proton/index.html). While not directly tested in MET-SWIM implementations, it is expected to work reliably based on the proven track record of other Qpid Proton language implementations.

### 13.2.5 DotNet

* **Qpid Proton DotNet** - .NET implementation from [Apache Qpid Proton](https://qpid.apache.org/proton/index.html), part of the same proven toolkit.

**Note**: At the time of writing, evidence of successful AMQP 1.0 implementations in other programming languages (such as Go, Ruby, PHP, or Rust) is limited. Organizations implementing MET-SWIM in these languages should verify compatibility with their chosen broker before deployment.

## 13.3 Filter Expression Examples

-- Filter by specific airports  
properties.icao\_location\_identifier IN ('EBBR', 'EDDF')  
  
-- Filter by country prefix  
properties.icao\_location\_identifier LIKE 'EB%'  
  
-- Filter by location and type  
properties.icao\_location\_identifier LIKE 'EB%'   
 AND properties.icao\_location\_type IN ('CTA', 'FIR', 'AD')  
  
-- Filter by geographic area  
geometry.type = 'Point'   
 AND geometry.coordinates.latitude BETWEEN 45 AND 50  
 AND geometry.coordinates.longitude BETWEEN 15.6 AND 20.7  
  
-- Filter by topic pattern  
topic LIKE '%.aviation.%'

# 14 AMQP Message Examples

## 14.1 METAR Message Example

# Message Addressing  
address: weather.aviation.metar  
  
# Header  
priority: 4  
  
# Message Properties  
subject: "DATA\_METAR\_EBBR\_NORMAL\_20250415120000"  
content-type: "application/xml"  
content-encoding: "gzip"  
absolute-expiry-time: 1744823400  
  
# Application Properties  
conformsTo: "https://eur-registry.swim.aero/services/eurocontrol-iwxxm-metar-speci-subscription-and-request-service-10"  
topic: "weather.aviation.metar"  
properties.datetime: "2025-04-15T12:00:00Z"  
properties.pubtime: "2025-04-15T12:02:00Z"  
properties.icao\_location\_identifier: "EBBR"  
properties.icao\_location\_type: "AD"  
geometry.type: "Point"  
geometry.coordinates.latitude: 50.9014  
geometry.coordinates.longitude: 4.4844  
links.count: 1  
links[0].href: "https://swim.example.com/filedb/DATA\_METAR\_EBBR\_NORMAL\_20250415120000.xml"  
links[0].rel: "canonical"  
links[0].type: "application/xml"  
  
# Payload  
# [Gzipped IWXXM XML content]

## 14.2 TAF Amendment Example

# Message Addressing  
address: weather.aviation.taf  
  
# Header  
priority: 5  
  
# Message Properties  
subject: "DATA\_TAF\_LOWS\_AMENDMENT\_20250401064500"  
content-type: "application/xml"  
content-encoding: "gzip"  
  
# Application Properties  
conformsTo: "https://eur-registry.swim.aero/services/eurocontrol-iwxxm-taf-subscription-and-request-service-10"  
topic: "weather.aviation.taf"  
properties.pubtime: "2025-04-01T06:45:00Z"  
properties.start\_datetime: "2025-04-01T06:00:00Z"  
properties.end\_datetime: "2025-04-02T06:00:00Z"  
properties.icao\_location\_identifier: "LOWS"  
properties.icao\_location\_type: "AD"  
links.count: 1  
links[0].href: "https://swim.example.com/filedb/DATA\_TAF\_LOWS\_AMENDMENT\_20250401064500.xml"  
links[0].rel: "update"  
links[0].type: "application/xml"

## 14.3 SIGMET Example

# Message Addressing  
address: weather.aviation.sigmet  
  
# Header  
priority: 7  
  
# Message Properties  
subject: "DATA\_SIGMET\_UDDD\_NORMAL\_20250415141000"  
content-type: "application/xml"  
content-encoding: "gzip"  
absolute-expiry-time: 1744813130  
  
# Application Properties  
conformsTo: "https://eur-registry.swim.aero/services/eurocontrol-iwxxm-sigmet-subscription-and-request-service-10"  
topic: "weather.aviation.sigmet"  
properties.pubtime: "2025-04-15T14:10:00Z"  
properties.start\_datetime: "2025-04-15T14:30:00Z"  
properties.end\_datetime: "2025-04-15T18:00:00Z"  
properties.icao\_location\_identifier: "UDDD"  
properties.icao\_location\_type: "FIR"  
geometry.type: "Bounding-box"  
geometry.coordinates.latitude-min: 38.83  
geometry.coordinates.latitude-max: 41.3  
geometry.coordinates.longitude-min: 43.47  
geometry.coordinates.longitude-max: 46.63  
properties.integrity.method: "sha512"  
properties.integrity.value: "826521fa4e298cc58388fc6a30e597f6..."  
links.count: 1  
links[0].href: "https://swim.example.com/filedb/DATA\_SIGMET\_UDDD\_NORMAL\_20250415141000.xml"  
links[0].rel: "canonical"  
links[0].type: "application/xml"  
  
# Payload  
# [Gzipped IWXXM XML content]

## 14.4 Multiple Links Example

This example shows how to handle messages with multiple links using the links.count property:

# Message Addressing  
address: weather.aviation.taf  
  
# Header  
priority: 5  
  
# Message Properties  
subject: "DATA\_TAF\_LPFL\_NORMAL\_20250415140000"  
content-type: "application/xml"  
content-encoding: "gzip"  
  
# Application Properties  
conformsTo: "https://eur-registry.swim.aero/services/eurocontrol-iwxxm-taf-subscription-and-request-service-10"  
topic: "weather.aviation.taf"  
properties.pubtime: "2025-04-15T14:00:00Z"  
properties.start\_datetime: "2025-04-15T15:00:00Z"  
properties.end\_datetime: "2025-04-16T00:00:00Z"  
properties.icao\_location\_identifier: "LPFL"  
properties.icao\_location\_type: "AD"  
geometry.type: "Point"  
geometry.coordinates.latitude: 39.45  
geometry.coordinates.longitude: -31.13  
links.count: 2  
links[0].href: "https://swim.example.com/filedb/DATA\_TAF\_LPFL\_NORMAL\_20250415140000.xml"  
links[0].rel: "canonical"  
links[0].type: "application/xml"  
links[1].href: "https://edr.swim.example.com/edr/collections/iwxxm-taf/locations/icao:LPFL?datetime=2025-04-15T15:00:00Z/2025-04-16T00:00:00Z"  
links[1].rel: "item"  
links[1].type: "application/zip"  
  
# Payload  
# [Gzipped IWXXM XML content]

## 14.5 Technical Message Example

# Message Addressing  
address: weather.aviation.metar # Same address as the data it pertains to  
  
# Header  
priority: 8 # High priority for technical messages  
  
# Message Properties  
subject: "TECHNICAL\_MAINTENANCE\_PAUSED\_20250415120000" # Structured pattern for technical messages  
content-type: "application/json"  
content-encoding: "identity"  
absolute-expiry-time: 1744909200 # 24 hours for maintenance messages  
  
# Application Properties  
conformsTo: "https://eur-registry.swim.aero/services/technical-message-service-10"  
topic: "weather.aviation.metar"  
properties.message\_type: "technical-message"  
  
# Payload  
{  
 "id": "770e8400-e29b-41d4-a716-446655440002",  
 "type": "maintenance-message",  
 "queue-status": "paused",  
 "timestamp": "2025-04-15T12:00:00Z",  
 "start": "2025-04-15T14:00:00Z",  
 "duration": 3600000,  
 "message": "Scheduled maintenance window - METAR service will be unavailable"  
}

# 15 Appendices

## 15.1 Appendix A: References to Example Implementations

* [IBL SWIM Demo Python Client](https://github.com/iblsoft/swimdemo)
* [IBL SWIM Weather Public Demo](https://swim.iblsoft.com/swimdemo/latest/SWIM-Weather-Public-Demonstration/)
* [DWD METAR in IWXXM 1.0](https://eur-registry.swim.aero/services/dwd-metar-iwxxm-10)
* [DWD TAF in IWXXM 1.0](https://eur-registry.swim.aero/services/dwd-taf-iwxxm-10)
* [DWD SIGMET in IWXXM 1.0](https://eur-registry.swim.aero/services/dwd-sigmet-iwxxm-10)

## 15.2 Appendix B: Property Extraction from IWXXM

This appendix provides comprehensive guidance for extracting AMQP application properties from IWXXM XML documents. IWXXM (ICAO Meteorological Information Exchange Model) documents follow a structured format defined by XML schemas that include multiple namespaces.

### 15.2.1 Required XML Namespaces

When processing IWXXM documents, XML namespaces must be properly detected and handled. **IWXXM versions and their namespace URIs change over time**, so implementations should use **version-agnostic namespace detection** rather than hardcoded URIs.

**Current IWXXM Schema Location**: <https://schemas.wmo.int/iwxxm/2023-1/>

**Dynamic Namespace Detection**:

Instead of hardcoding specific version URIs, detect namespaces dynamically by their common prefixes:

| Namespace Family | Common Prefix Pattern | Purpose |
| --- | --- | --- |
| IWXXM | http://icao.int/iwxxm/ or https://schemas.wmo.int/iwxxm/ | IWXXM meteorological reports |
| AIXM | http://www.aixm.aero/schema/ | Aeronautical information (airports, airspace) |
| GML | http://www.opengis.net/gml/ | Geographic markup language |

**Implementation Approach for Namespace Detection**:

1. **Parse the XML document** and access the root element
2. **Extract all namespace declarations** from the document (typically available through the XML parser’s namespace map or similar functionality)
3. **Iterate through namespace URIs** and match them against the common prefix patterns shown in the table above
4. **Build a namespace mapping** that maps logical prefixes (iwxxm, aixm, gml, etc.) to the actual namespace URIs found in the document
5. **Use this mapping** when constructing XPath expressions for data extraction

**Python Implementation**: The [IBL swimdemo iwxxm\_utils.py](https://github.com/iblsoft/swimdemo/blob/main/iwxxm_utils.py) provides a working example of this approach.

**Legacy vs. Current Namespace Examples**:

<!-- Legacy IWXXM (pre-2023) -->  
xmlns:iwxxm="http://icao.int/iwxxm/2.1"  
xmlns:aixm="http://www.aixm.aero/schema/5.1.1"  
xmlns:gml="http://www.opengis.net/gml/3.2"  
  
<!-- Current IWXXM (2023+) -->  
xmlns:iwxxm="https://schemas.wmo.int/iwxxm/2023-1"  
xmlns:aixm="http://www.aixm.aero/schema/5.1.1"  
xmlns:gml="http://www.opengis.net/gml/3.2"

**Note**: Always detect namespaces dynamically from the actual XML document rather than assuming specific versions. This ensures compatibility with both legacy and current IWXXM implementations.

### 15.2.2 Document Structure Overview

IWXXM documents have different root elements depending on the report type:

* **METAR/SPECI**: <iwxxm:METAR>
* **TAF**: <iwxxm:TAF>
* **SIGMET**: <iwxxm:SIGMET>

Each document contains:

1. **Report metadata** (issue time, status, location references)
2. **Validity period** (for TAF and SIGMET)
3. **Observation time** (for METAR/SPECI)
4. **Location information** (aerodrome or airspace references)
5. **Meteorological content** (actual weather observations/forecasts)

### 15.2.3 Property Extraction Reference

| AMQP Property | Report Types | XPath Expression | Description | Example Value |
| --- | --- | --- | --- | --- |
| properties.pubtime | ALL | /iwxxm:\*/iwxxm:issueTime | Issue/publication time from root element | 2025-04-15T14:10:00Z |
| properties.datetime | METAR, SPECI | /iwxxm:METAR/iwxxm:observationTime | Observation time | 2025-03-31T03:00:00Z |
| properties.start\_datetime | TAF, SIGMET | /iwxxm:\*/iwxxm:validPeriod/gml:beginPosition | Start of validity period | 2025-04-15T14:30:00Z |
| properties.end\_datetime | TAF, SIGMET | /iwxxm:\*/iwxxm:validPeriod/gml:endPosition | End of validity period | 2025-04-15T18:00:00Z |
| properties.icao\_location\_identifier | METAR, SPECI, TAF | /iwxxm:\*/iwxxm:aerodrome/aixm:AirportHeliport/aixm:locationIndicatorICAO | Aerodrome ICAO code | EBBR |
| properties.icao\_location\_identifier | SIGMET | /iwxxm:SIGMET/iwxxm:issuingAirTrafficServicesRegion/aixm:Airspace/aixm:designator | Airspace designator | UDDD |
| properties.icao\_location\_type | METAR, SPECI, TAF | "AD" | Always “AD” for aerodrome reports | AD |
| properties.icao\_location\_type | SIGMET | /iwxxm:SIGMET/iwxxm:issuingAirTrafficServicesRegion/aixm:Airspace/aixm:type | Airspace type | FIR, UIR, CTA, OTHER:FIR\_UIR |

### 15.2.4 Detailed Extraction Guidelines

#### 15.2.4.1 Publication Time (properties.pubtime)

**All Report Types**: The issue time is always located at the root level of the IWXXM document.

/iwxxm:METAR/iwxxm:issueTime  
/iwxxm:TAF/iwxxm:issueTime   
/iwxxm:SIGMET/iwxxm:issueTime

**Example XML Structure**:

<iwxxm:METAR>  
 <iwxxm:issueTime>2025-04-15T14:10:00Z</iwxxm:issueTime>  
 <!-- ... rest of document ... -->  
</iwxxm:METAR>

#### 15.2.4.2 Observation Time (properties.datetime)

**METAR/SPECI Only**: Located directly under the root element.

/iwxxm:METAR/iwxxm:observationTime

**Example XML Structure**:

<iwxxm:METAR>  
 <iwxxm:issueTime>2025-04-15T14:10:00Z</iwxxm:issueTime>  
 <iwxxm:observationTime>2025-04-15T14:00:00Z</iwxxm:observationTime>  
 <!-- ... rest of document ... -->  
</iwxxm:METAR>

#### 15.2.4.3 Validity Period (properties.start\_datetime, properties.end\_datetime)

**TAF and SIGMET**: The validity period uses GML time elements.

/iwxxm:TAF/iwxxm:validPeriod/gml:beginPosition  
/iwxxm:TAF/iwxxm:validPeriod/gml:endPosition  
/iwxxm:SIGMET/iwxxm:validPeriod/gml:beginPosition  
/iwxxm:SIGMET/iwxxm:validPeriod/gml:endPosition

**Example XML Structure**:

<iwxxm:TAF>  
 <iwxxm:issueTime>2025-04-15T14:10:00Z</iwxxm:issueTime>  
 <iwxxm:validPeriod>  
 <gml:beginPosition>2025-04-15T15:00:00Z</gml:beginPosition>  
 <gml:endPosition>2025-04-16T15:00:00Z</gml:endPosition>  
 </iwxxm:validPeriod>  
 <!-- ... rest of document ... -->  
</iwxxm:TAF>

#### 15.2.4.4 ICAO Location Identifier and Type

**For Aerodrome Reports (METAR, SPECI, TAF)**:

The aerodrome reference may be either embedded or referenced via xlink:href.

**Embedded aerodrome**:

/iwxxm:\*/iwxxm:aerodrome/aixm:AirportHeliport/aixm:locationIndicatorICAO

**Referenced aerodrome** (when xlink:href is used):

/iwxxm:\*/iwxxm:aerodrome/@xlink:href

When using xlink:href, the ICAO code is typically embedded in the URI (e.g., urn:uuid:81e47548-9f00-4970-b24e-d53bf4986257#EBBR).

**Example XML Structure (Embedded)**:

<iwxxm:METAR>  
 <iwxxm:aerodrome>  
 <aixm:AirportHeliport gml:id="aerodrome-EBBR">  
 <aixm:timeSlice>  
 <aixm:AirportHeliportTimeSlice gml:id="aerodrome-EBBR-ts">  
 <gml:validTime>  
 <gml:TimePeriod gml:id="aerodrome-EBBR-ts-vt">  
 <gml:beginPosition indeterminatePosition="before"/>  
 <gml:endPosition indeterminatePosition="after"/>  
 </gml:TimePeriod>  
 </gml:validTime>  
 <aixm:interpretation>BASELINE</aixm:interpretation>  
 <aixm:locationIndicatorICAO>EBBR</aixm:locationIndicatorICAO>  
 </aixm:AirportHeliportTimeSlice>  
 </aixm:timeSlice>  
 </aixm:AirportHeliport>  
 </iwxxm:aerodrome>  
 <!-- ... rest of document ... -->  
</iwxxm:METAR>

**Example XML Structure (Referenced)**:

<iwxxm:METAR>  
 <iwxxm:aerodrome xlink:href="urn:uuid:81e47548-9f00-4970-b24e-d53bf4986257#EBBR"/>  
 <!-- ... rest of document ... -->  
</iwxxm:METAR>

**For Airspace Reports (SIGMET)**:

/iwxxm:SIGMET/iwxxm:issuingAirTrafficServicesRegion/aixm:Airspace/aixm:designator  
/iwxxm:SIGMET/iwxxm:issuingAirTrafficServicesRegion/aixm:Airspace/aixm:type

**Example XML Structure**:

<iwxxm:SIGMET>  
 <iwxxm:issuingAirTrafficServicesRegion>  
 <aixm:Airspace gml:id="fir-UDDD">  
 <aixm:timeSlice>  
 <aixm:AirspaceTimeSlice gml:id="fir-UDDD-ts">  
 <gml:validTime>  
 <gml:TimePeriod gml:id="fir-UDDD-ts-vt">  
 <gml:beginPosition indeterminatePosition="before"/>  
 <gml:endPosition indeterminatePosition="after"/>  
 </gml:TimePeriod>  
 </gml:validTime>  
 <aixm:interpretation>BASELINE</aixm:interpretation>  
 <aixm:type>FIR</aixm:type>  
 <aixm:designator>UDDD</aixm:designator>  
 </aixm:AirspaceTimeSlice>  
 </aixm:timeSlice>  
 </aixm:Airspace>  
 </iwxxm:issuingAirTrafficServicesRegion>  
 <!-- ... rest of document ... -->  
</iwxxm:SIGMET>

### 15.2.5 Report Status Extraction

For the AMQP subject property, extract the report status from:

/iwxxm:\*/@reportStatus

Common values:

* NORMAL - Regular report
* AMENDMENT - Amendment to previous report
* CORRECTION - Correction to previous report

**Example XML Structure**:

<iwxxm:TAF reportStatus="AMENDMENT">  
 <!-- ... document content ... -->  
</iwxxm:TAF>

### 15.2.6 Geometry Information Extraction

**For Point Locations (Aerodromes)**:

/iwxxm:\*/iwxxm:aerodrome/aixm:AirportHeliport/aixm:ARP/aixm:ElevatedPoint/gml:pos

The gml:pos element contains space-separated latitude and longitude values.

**For Area Locations (SIGMET)**:

SIGMET documents may contain complex geometry. For bounding box extraction:

/iwxxm:SIGMET/iwxxm:analysis/iwxxm:SigmetAnalysis/iwxxm:phenomenonGeometry//gml:posList