AMQP Message Structure in MET-SWIM (CP1)

Table of Contents

In August 2025 the AMQP guidance was split into two parts: *CP1* and *Next*

* The original working copy of this document is maintained in Markdown syntax: [MET-SWIM-AMQP-Guidance-CP1.md](https://github.com/iblsoft/swimdemo/blob/main/MET-SWIM-AMQP-Guidance-CP1.md).
* Properties that go beyond the CP1 MET-SWIM Service Definitions are maintained separately in [MET-SWIM-AMQP-Guidance-Next.md](https://github.com/iblsoft/swimdemo/blob/main/MET-SWIM-AMQP-Guidance-Next.md).

Members of EUROCONTROL MET3SG Task Team on Service Architecture can submit pull requests towards the Markdown documents or comment on the .DOCX version on MET3SG Sharepoint.

# 1 Version History

| Version | Date | Changes |
| --- | --- | --- |
| 1.0-draft | 2025-06 | Initial draft. |
| 1.0-draft2 | 2025-08 | Incorporated feedback, simplified, split into “CP1” and “Next” for properties beyond CP1. |

# 2 Introduction

This 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 distribution using AMQP 1.0 protocol in line with the SWIM Service definitions.

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

## 2.1 Objectives

* Ensure interoperability between different MET-SWIM implementations
* Standardize AMQP message properties for meteorological data exchange
* Enable efficient filtering of messages
* Enable routing of messages to downstream systems based on typical information like report type or aerodrome/FIR code.
* Provide forward compatibility for future meteorological data types beyond CP1

## 2.2 Scope (for CP1)

This specification currently covers the following meteorological report types that are covered by CP1 SWIM Service Definitions in IWXXM format:

* METAR
* SPECI
* TAF
* SIGMET

## 2.3 WMO WIS 2.0 Notification Messages Relationship

This specification takes inspiration from the [WMO WIS 2.0 Notification Message Standard](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html) while being tailored to the AMQP property syntax and specific requirements of MET-SWIM and the aviation meteorological community.

The WNM notification specification also guided the design of the notification messages in [OGC API - Environmental Data Retrieval (EDR) Part 2: Publish-Subscribe workflow](https://docs.ogc.org/is/23-057r1/23-057r1.html) which is used for example by the UK MetOffice’s QVA API (Quantitative Volcanic Ash) and is planned to be used by SADIS API.

## 2.4 Authors

Editors:

* Michal Weis (IBL)
  + The original proposal of mapping WIS 2 WNM notification metadata in JSON to AMQP application properties.
* Boris Burger (IBL)
  + Further alignment with AMQP, WNM, and incorporating feedback.
  + Relationship of AMQP properties to IWXXM data model.

Feedback provided by:

* Dario di Crescenzo (EUROCONTROL)
* Michael Pichler (Austro Control)
* Tiaan Wessels (Netsys)
* Jürgen Schulze (met.no)
* Florian Dobener (DWD)

# 3 References

* [EU Common Project Regulation (CP1)](https://eur-lex.europa.eu/eli/reg_impl/2021/116/oj/eng)
* [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)
* [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)
* [OGC API - Environmental Data Retrieval (EDR)](https://ogcapi.ogc.org/edr/)
* [OGC API - Environmental Data Retrieval (EDR) Part 2: Publish-Subscribe workflow](https://docs.ogc.org/is/23-057r1/23-057r1.html)
* [AMQP Filter Expressions 1.0](https://docs.oasis-open.org/amqp/filtex/v1.0/filtex-v1.0.html)
* [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)

# 4 AMQP 1.0 Message Structure

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

1. **Message Addressing** - Destination address using a hierarchical structure
2. **Message Header** - Priority and TTL (time-to-live) 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).

# 5 Message Addressing

## 5.1 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

## 5.2 Examples

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

## 5.3 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 available in the popular AMQP brokers, but is not guaranteed in general.

**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

### 5.3.1 Broker Compatibility

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

#### 5.3.1.1 Apache ActiveMQ Artemis

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

#### 5.3.1.2 Apache Qpid Proton

* Supports both \* and # wildcards

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

* There is some support for \* and #
* RabbitMQ supports the wildcards for routing from topic exchanges to queues. It does not, however, allow for automatic creation of queues from a topic. The users would need to create a queue and its binding themselves.

# 6 AMQP Transport Header

## 6.1 Priority

The priority field (0-9, where 9 is highest) SHALL be set according to the operational importance. For this version of the specification the exact priorities are not defined, however:

* SPECI should have higher priority than METAR
* TAF amendments should have higher priority than TAF
* SIGMET should have the highest priority

Example priorities:

| Report Type | Priority | Rationale |
| --- | --- | --- |
| SIGMET | 7 | Highest priority for safety-critical information |
| SPECI | 6 | Special reports indicating significant changes |
| TAF AMD | 6 | Significant change to the original forecast |
| TAF | 5 | Forecast information with medium priority |
| METAR | 4 | Routine observations |

# 7 AMQP Message 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 documentation for the AMQP headers specifically is in [OASIS 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).
* See also [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 more details.

## 7.1 subject (MANDATORY)

The subject SHALL represent the type of data transmitted, analogously to the WMO WIS 2.0 concept of topics, and equal to the *source address*. This is useful for disambiguation of data when using wildcard source address subscriptions.

Example:

weather.aviation.metar

**Note:** In earlier versions of this specification it was proposed that the subject string should hold the report type (METAR, SPECI, TAF, SIGMET), aerodrome or airspace ICAO code, issue time of the report, and so on. However, this was creating a duplicate identification and filtering mechanism competing with the application properties. Application properties along with the SQL-like filter expressions are much better suited for filtering purposes compared to wildcard patterns applied to subject strings.

## 7.2 content-type (MANDATORY)

Indicates the MIME type of the payload and SHALL be set to application/xml for the IWXXM XML data.

## 7.3 content-encoding (CONDITIONAL)

By default the payloads are not compressed, and in this case the content-encoding property can be left out. However, when compression is applied to the payload, then the content-encoding is mandatory.

The MET-SWIM service AMQP implementations and clients SHALL support the following content-encodings from [IANA HTTP Content Coding Registry|https://www.iana.org/assignments/http-parameters/http-parameters.xhtml]:

* gzip - for compressed data. This is the most widely supported compressed content-encoding in HTTP.
* identity - for uncompressed data (default if omitted).

## 7.4 absolute-expiry-time (OPTIONAL)

Unix timestamp indicating when the message should expire in the broker and from the durable queues.

The absolute-expiry-time tells the broker when it should discard a message. On the other hand the ttl field in AMQP transport header (relative number in milliseconds) is typically calculated using a difference between the absolute-expiry-time and the time when the message was created. The absolute-expiry-time stays constant during the message’s lifetime, whereas ttl can be progressively decreased by AMQP intermediaries.

When absolute-expiry-time is not set, the message will “live” in durable queues until a client receives the message. Brokers will usually have some pre-configured threshold of maximum ttl to avoid resource exhaustion if clients are not picking up messages from durable queues.

If a MET-SWIM AMQP implementation sets the absolute-expiry-time it SHOULD set it high enough to:

* give the clients a chance to receive the messages during shorted communication outages or client maintenance periods.
* keep the messages long enough to survive the AMQP server maintenance windows or outages.

Example: AMQP message creation + 12 hours

Note: Originally it was proposed to set the absolute-expiry-time depending on the report type, e.g. 3h for METAR, 12h for TAF, 24h for SIGMET. However, in general there can be varying purposes of why the client needs to be subscribed to the data:

* One client might be interested only in the latest operationally useful data.
* Another client might be a climatological database that needs all the reports no matter how old. In this case loosing observations due to a 3h communication outage does not seem reasonable.

## 7.5 creation-time (OPTIONAL)

This is the Unix timestamp indicating when the AMQP message was originally created. It mostly servers an informational purpose in AMQP, and is used in ttl calculation in some of the brokers.

# 8 AMQP Application Properties

[AMQP application property](https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#type-application-properties) names and meanings are not regulated by the AMQP specification. They are custom defined by individual services that use AMQP as its notification layer. The application properties are the most flexible mechanism for identifying the data sent using AMQP and enable server-side filtering using [AMQP Filter Expressions 1.0](https://docs.oasis-open.org/amqp/filtex/v1.0/filtex-v1.0.html).

Properties in sections below are divided into categories based on their usage requirements:

* *mandatory*: need to be present at all times,
* *conditional*: are mandatory under stated conditions,
* *optional*: their presence is left to the implementation, but when included they SHALL adhere to this specification.

**Note:** Earlier revisions of this guidance used a properties. prefix to stay close to the *WIS 2.0 Notification Message (WNM)* standard. However:

1. The usage of the properties object in WNM is only due to the WIS 2.0 notification message conforming to the GeoJSON formatting ([RFC 7946](https://datatracker.ietf.org/doc/html/rfc7946)).
2. GeoJSON specification requires that all custom properties must be stored in a properties sub-object, rather than in the top-level object that is reserved for feature type declaration and geometry.
3. In AMQP this is not necessary, because the direct equivalent of the GeoJSON custom properties are the AMQP application properties, so there is no need to use any prefix.
4. See also the [WNM message](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html#_wis2_notification_message_examples) example from the WIS 2.0 specification.

**TODO:** is this a good decision, or should we go back to using the "properties." prefix to have a more 1:1 correspondence to the WNM JSON?

## 8.1 Examples of AMQP Filter Expressions applied to Application Properties

-- Filter by specific airports  
icao\_location\_identifier IN ('EBBR', 'EDDF')  
  
-- Filter by country prefix  
icao\_location\_identifier LIKE 'EB%'  
  
-- Filter by location and type  
icao\_location\_identifier LIKE 'EB%'   
 AND icao\_location\_type IN ('CTA', 'FIR', 'AD')  
  
-- Filter by topic pattern  
subject LIKE 'weather.aviation.%'

## 8.2 Mandatory Application Properties

These properties MUST be present in all meteorological data messages.

### 8.2.1 report\_status (mandatory)

One of the following values:

* NORMAL - Regular report
* AMENDMENT - Amendment of a previous report
* CORRECTION - Correction of a previous report

This is directly based on the IWXXM reportStatus attribute.

### 8.2.2 icao\_location\_identifier (mandatory)

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.

icao\_location\_identifier: "EBBR"

### 8.2.3 icao\_location\_type (mandatory)

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>

### 8.2.4 conformsTo (conditional)

If there is a service definition the data conforms to (e.g. the CP1 compliant SWIM Service Definitions), this field SHOULD link to the service definition. The field 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

## 8.3 Temporal Properties

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

### 8.3.1 issue\_time (mandatory)

[RFC 3339](https://datatracker.ietf.org/doc/html/rfc3339) formatted publication/issue time extracted from iwxxm:issueTime:

issue\_time: "2025-04-15T14:10:00Z"

This property is mandatory for all the CP1 message types.

**Note:** In previous proposals this was called pubtime in relation to WIS 2.0 WNP publication time (properties.pubtime). However, the meaning of publication time in WNM is the time when the message was originally sent, which is closer to the meaning of the creation-time in the AMQP transport header.

## 8.4 Conditional Properties - Temporal

These properties are required based on the message type:

### 8.4.1 datetime (conditional)

For METAR/SPECI only - observation time in [RFC 3339](https://datatracker.ietf.org/doc/html/rfc3339) format from iwxxm:observationTime:

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

**Note:** Corresponds to properties.datetime in WIS 2.0 Notification Messages, see [Properties / Temporal description](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html#_1_12_properties_temporal_description) in the WNM specification.

### 8.4.2 start\_datetime (conditional)

For TAF/SIGMET - start of validity period in [RFC 3339](https://datatracker.ietf.org/doc/html/rfc3339) format from iwxxm:validPeriod:

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

**Note:** start\_datetime and end\_datetime properties correspond to the equivalently named properties in the WIS 2.0 Notification Messages. See [Properties / Temporal description](https://wmo-im.github.io/wis2-notification-message/standard/wis2-notification-message-STABLE.html#_1_12_properties_temporal_description) in the WNM specification.

### 8.4.3 end\_datetime (conditional)

For TAF/SIGMET - end of validity period in [RFC 3339](https://datatracker.ietf.org/doc/html/rfc3339) format:

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

## 8.5 AMQP with Message Security

SWIM TI Yellow Profile 2.0 section 3.1.1.12 defines Service Interface Binding of type “AMQP Messaging with Message Security” that enables detection of data corruption during transmission and ensures data authenticity. This is by leveraging the [S/MIME 4.0 (RFC 8551)](https://datatracker.ietf.org/doc/html/rfc8551) standard used in emails.

**Note:** Our previous proposal was to adopt integrity methods from WIS 2.0 WNM specification, but since Yellow Profile 2.0 endorses S/MIME 4.0 there is no need to copy the integrity mechanisms from WIS 2.0.

**TODO:** Provide more guidance on using S/MIME 4.0.

# 9 Message Payload

For IWXXM XML documents, the payload SHALL contain the complete IWXXM XML document representing one METAR, SPECI, TAF or SIGMET report. XML documents using

* 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.

# 10 Broker and Client Library Compatibility

## 10.1 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 | partial | ✓ | Requires AMQP 1.0 plugin enabled |
| Apache Qpid Broker-J | ✓ | ✓ | Apache’s AMQP 1.0 broker implementation |

## 10.2 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:

**Note:** This section needs to be expanded based on experience of a wider range of organizations. Qpid Proton C++ and Python APIs are known to work correctly.

### 10.2.1 JavaScript/Node.js

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

### 10.2.2 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.

### 10.2.3 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.

### 10.2.4 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.

### 10.2.5 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.

# 11 AMQP Message Examples

## 11.1 METAR Message Example

# Message Addressing  
address: weather.aviation.metar  
  
# Header  
priority: 4  
  
# Message Properties  
subject: "weather.aviation.metar"  
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"  
issue\_time: "2025-04-15T12:02:00Z"  
datetime: "2025-04-15T12:00:00Z"  
icao\_location\_identifier: "EBBR"  
icao\_location\_type: "AD"  
  
# Payload  
# [Gzipped IWXXM XML content]

## 11.2 TAF Amendment Example

# Message Addressing  
address: weather.aviation.taf  
  
# Header  
priority: 5  
  
# Message Properties  
subject: "weather.aviation.taf"  
content-type: "application/xml"  
content-encoding: "gzip"  
  
# Application Properties  
conformsTo: "https://eur-registry.swim.aero/services/eurocontrol-iwxxm-taf-subscription-and-request-service-10"  
issue\_time: "2025-04-01T06:45:00Z"  
start\_datetime: "2025-04-01T06:00:00Z"  
end\_datetime: "2025-04-02T06:00:00Z"  
icao\_location\_identifier: "LOWS"  
icao\_location\_type: "AD"

## 11.3 SIGMET Example

# Message Addressing  
address: weather.aviation.sigmet  
  
# Header  
priority: 7  
  
# Message Properties  
subject: "weather.aviation.sigmet"  
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"  
issue\_time: "2025-04-15T14:10:00Z"  
start\_datetime: "2025-04-15T14:30:00Z"  
end\_datetime: "2025-04-15T18:00:00Z"  
icao\_location\_identifier: "UDDD"  
icao\_location\_type: "FIR"  
  
# Payload  
# [Gzipped IWXXM XML content]

# 12 Appendices

## 12.1 Appendix A: 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.

### 12.1.1 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.

### 12.1.2 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)

### 12.1.3 Property Extraction Reference

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