Íslenskir staðlar

ÍST TS 316:2025

Entry into force 21.03.2025

ICS 35.240

Vinnustofusamþykkt - IOBWS

útgáfa 3 tæknilegar viðmiðanir

Workshop Agreement - IOBWS

version 3 Technical Guidelines

Íslenskir staðlar

ÍST TS 316:2025

Participants in ÍST Workshop VH-7 during the development of ÍST TS 316_2025 IOBWS 3.1 Technical Guidelines.

Name	Company	Name	Company
Valdimar Valdemarsson	Origo	Daníel Sveinsson	Arion banki
Halldór Vagn Hreinsson	Íslandsbanki	Sigurvin Frank Garðarsson	Íslandsbanki
Jóhannes Þór Ágústarson	Íslandsbanki	Steinar Logi Sigþórsson	Arion banki
Kristinn Stefánsson	Arion banki	Guðmundur Andri Hjálmarsson	RB
Kristján Theódór Sigurðsson	Arion banki	Þórný Pétursdóttir	RB
Björn Ingi Björnsson	Arion banki		

2nd edition

Table of contents

ÍST TS 316:2025

Foreword

This ÍST Technical Specification was developed in accordance with “ÍST Reglur um tækniforskriftir, tækniskýrslur og vinnustofusamþykktir” (*e.* IST rules on Technical Specifications, Technical Reports and Workshop Agreements). The TS (Technical Specification) was prepared by working groups VH-7 and VH-8, with input from VH-1 and VH-2 under the technical committee TN-FMÞ (The Technical Committee on Financial Services) that operates within FUT (Sector committee for ICT standardisation) following a public call for participation within TN-FMÞ. The final draft was sent to the TN-FMÞ on the 2025-03-06 and approved by correspondence on the 2025-03-06. The text of ÍST TS-316 was submitted to IST for publication on 2025-03-06.

The document “ÍST TS 316_2025 IOBWS 3.1 Technical Guidelines” is the source of this rendition, and versions of that document will be used for future errata and clarifications per the procedures laid out by the working groups.

The work on the ÍST TS was primarily funded by Arion Banki, Íslandsbanki and Landsbankinn. It is the result of the original workgroup TN-FMÞ-VH-8 and the maintainance efforts of TN-FMÞ-VH-7.

The Technical Committee’s participants have made every effort to ensure the reliability and accuracy of the technical and non-technical content of ÍST TS-316, but this does not guarantee, either explicitly or implicitly, its correctness. Users of ÍST TS-316 should be aware that neither the TN-FMÞ nor ÍST can be held liable for damages or losses of any kind whatsoever which may arise from its application. Users of ÍST TS-316 do so on their own responsibility and at their own risk.

Introduction

This Technical Specifications (TS) presents aspects of the framework for maintaining and implementing API service in version 3.x of the Icelandic Online Banking Services (IOBWS).

Previous versions of IOBWS, released in 2007 and 2013 respectively, used the most recent OASIS SOAP standards at the time, to define common web service interfaces for the Icelandic commercial and savings banks. This enabled software vendors, enterprises and service providers to integrate their accounting, payment, and information systems with the bank's services, to act on behalf of the customers and with full access to their data. Further examples of usage are discussed in the section on Use Cases.

When initiating work on the previous versions, the participants in the TN-FMÞ reviewed existing and emerging specifications in the global or mostly European financial industry. None were deemed a good fit at the time for local adaptation, as they reflected inherent the legacy in inter-bank communications outside of Iceland.

The Open Banking regulation in the UK and the PSD2 regulation issued by the European Parliament has triggered initiatives to standardize access to payment functionality and account information, on behalf of customers by third parties. One such effort, the NextGenPSD2 Framework developed by the Berlin Group [4], has met broad acceptance in the EEA. The data model references ISO 20022 [1] and is close enough to the direction of the Icelandic market to make it relatively straightforward to adapt it as the new base for the IOBWS, instead of continuing to maintain an independent linage of API specifications.

Another goal achieved by adopting the NextGenPSD2 Framework is the transition from SOAP to a REST-like API defined by a version of the Open API Specification [2]. Along with support for modern authentication and authorization standards, this addresses some of the perceived complexity in adapting IOBWS to use cases, platforms and programming languages that have come to the fore after the release of the previous IOBWS versions.

ÍST TS-316 includes information on common implementation details and cross-cutting concerns related to the technical specifications that together form IOBWS version 3. It is intended to be an evolving document with each minor version issued as a new Technical Specification. The process for this is described in section 3.

ÍST TS 316:2025

1 Scope

The document addresses specifics related to authentication and authorization, as well as idempotency. Also defined are the processes for maintaining the overall framework of IOBWS version 3.0, documents as well as the associated git repository. Under scope are the following technical specification documents:

- TS 312:2021 Currency
- TS 310:2022 Domestic payments and deposits
- TS 311:2021 Debit and credit cards details and statements
- TS 315 Claims
- TS 314:2021 Documents
- TS 313:2021 Foreign Payments

2 Normative references, definitions and data elements

2.1 Normative references

The following documents are referred to in ÍST TS-316 as part of their content constitutes the requirements of this document. Only the edition cited applies if newer editions exist.
NextGenPSD2 v1.3.8. *The Berlin Group NextGenPSD2 Access to Account Framework*.

OpenAPI v3.0.1. The OpenAPI Specification (OAS) by the OpenAPI Initiative, a Linux Foundation Collaborative Project.

2.2 Terms and definitions

- **Berlin Group** is a pan-European payments interoperability standards and harmonisation initiative with the primary objective of defining open and common scheme- and processor-independent standards in the interbanking domain between Creditor Bank (Acquirer) and Debtor Bank (Issuer), complementing the work carried out by e.g. the European Payments Council. As such, the Berlin Group has been established as a pure technical standardisation body, focusing on detailed technical and organisational requirements to achieve this primary objective.
- **Electronic IDentification, Authentication and trust Services** (**eIDAS**) refers to regulation 910/2014 [7], which replaced previous directive 1999/93/EC. It was introduced to Iceland law through act no. 2019/55 [5].
- **The OpenAPI Specification** (**OAS**) defines a programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic.

3 Ongoing Maintainance of IOBWS version 3

The rules outlined in this section are accepted by the participants in TN-FMÞ as the process for further development of the documents and specifications that form IOBWS version 3.

They establish the Github Git repository IST-FUT-FMTH created and supervised by the Icelandic Standards Council as the master source for the associated OpenAPI specifications, as well as those workgroup agreements and technical specifications that have been codified into Markdown markup language and stored in the repository. All updates to the main branch shall be done through pull requests and the merging of said pull requests after review.

The participants in TN-FMÞ agree that workgroup TN-FMÞ-VH-7 is in charge of monitoring submitted issues made to the repository when they fall outside the permit of other regular workgroups. VH-7 will regularly review those issues, give feedback, close or potentially classify them as part of larger initiatives or projects. TN-FMÞ-VH-7 will evaluate if changes submitted in the form of pull requests are fit to be accepted into the repository and when or if they warrant a patch or minor releases to the overall specification. Versioning will adhere to the Semantic Versioning[6] scheme but each minor release will require a Workgroup agreement under the “ÍST reglur” referenced above.

Should TN-FMÞ form future workgroups around larger initiatives, they will adhere to the same rules for change management. Specific development branches will be created for the work and the groups be in charge of accepting individual pull requests into their branch. The acceptance or rather a pull-request merge into the main branch will be handled by VH-7 based on the final approval of the initiative in question. A workgroup agreement or technical specification update will be required as part of such approval, when not otherwise decided by VH-7.

Further notes on contributing and details on how to adhere to these rules are included or referenced in the ReadMe Markdown document at the root of the repository and will be updated as deemed necessary by VH-7.

4 Authentication Use Cases and Requirements

The API specifications for IOBWS reference OAuth2 based authorization, with the NextGenPSD2 ancestry of parts of the specification, occasionally showing through as references to consents. It is the intention here to further elaborate on the ways the most common use cases should be handled as the common dominator among implementors and consumers of the APIs.

It is established here that the usage of “Búnaðarskilríki” issued under Fullgilt Auðkenni as the current gold standard for authentication, will continue to be supported. They will not require the usage of usernames and passwords as in the previous IOBWS specifications, though implementers may use other standard mechanisms such as client IDs and secrets to distinquish between different roles (see Usection **??**).

Additionally, OpenID Code Flow with PKCE will be part of the common support to handle the various scenarios.

4.1 Main Use Cases section **??**

To harmonize technical expectations, some basic use cases are considered and the acceptance criteria should guide implementors towards selecting the correct solution.

4.1.1 Centralized Financial System

> As a **Financial System**, I want to connect to IOBWS 3.0 services so that I can e.g. manage Claims, initiate Payments, and fetch Account transactions in batches or directly on behalf of users.

Acceptance criteria:

1. Support for Búnaðarskilríki issued under Fullgilt Auðkenni, for authenticating and authorizing centralized software to act on behalf of organizational units.
2. Support for OIDC and, code flow with PKCE as the common denominator.
3. Support for online scenarios, where the organization authenticates the interactive employee that instigates the action.

4.1.2 On-premise System

> As **a user of an on-premise Financial System**, I want to be able to authorize the system to connect to IOBWS 3.0 services and manage Claims, initiate Payments and fetch Account transactions on my behalf in non-interactive sessions.

Acceptance criteria:

1. Support for OIDC and OAuth 2.0, code flow with offline_access, using MTLS to identify the client/server using Búnaðarskilríki issued under Fullgilt Auðkenni.

4.1.3 On-premise employee

> As a **company employee** I want to e.g. initiate payment instructions, create claims and interact with IOBWS 3.0 so that I can manage my day-to-day activities through e.g. the company ERP system.

Acceptance criteria:

1. Support for OIDC and OAuth 2.0, code flow with PKCE as the common denominator.
2. Support for user authentication with Qualified Certificates.

4.1.4 Financial Services

> As e.g. **an independent Accounting firm** offering services to multiple clients, I want to be able to access their accounts and products through IOBWS 3.0, so I can manage their financials and accounting.

Acceptance criteria:

1. Support for assuming multiple identities.
2. Support for OIDC and OAuth 2.0, code flow with PKCE as the common denominator.
3. Support for user authentication with Qualified Certificates.
4. The scopes should be known, based on the endpoints defined in IOBWS 3.0.

4.1.5 Financial Software as a Service

> As the IOBWS 3.0 **customer of a bank**, I want to be able to authorize SaaS software hosted in public clouds to act on my behalf, so I can allow the service to manage my financials and the products I have access to such as Claims.

Acceptance criteria:

1. This should address the scenario where companies offering e.g. Dynamics 365 or more custom apps need to act on behalf of bank customers.
2. Support for OIDC and OAuth 2.0, code flow with PKCE as the common denominator.
3. Support for user authentication with Qualified Certificates.

4.1.6 Software Vendor

> As a **Software Vendor** providing Custom, COTS, or SaaS applications that my clients use to access IOBWS 3.0, I want to be able to target common authentication behaviour as part of the technical standard offered by all the banks, so that I do not have to implement and test against multiple subtly different endpoints.

Acceptance criteria:

1. There exist code samples that show how to connect using common platforms and frameworks.
2. The possible variations between banks do not affect the protocol exchanges between the client, authorization server, and API endpoint.
3. Possible variations in methods that still are offered by more than one bank are made part of the standard, as long as a common fallback exists.

4.1.7 User of open endpoints

> As a **Consumer of open services** such as currency data, I want my system to be able to interact with the endpoints without authentication but identify my client as to Acceptance criteria:

4.1.8 Enterprise with the Claim Collection Agency role logs in for the first time

> As a **Claim Collection Agency**, I want my system to be able to login into the system and separate my authentication as a secondary collection agency vs. my use as a primary claims collector.

Acceptance criteria:

1. When I log in as a secondary collection role, I can choose to identify using a client ID that is related to that role.
2. When I log in as the parent enterprise to create claims as a primary claims collector, I can choose to identify using a client ID that is related to that role.

4.1.9 Claim Collection Agency

> As a **Claim Collection Agency**, I want my system to be able to interact with the endpoints to manipulate claims whose status is in the secondary collection and transferred to a claim template in my ownership.

Acceptance criteria:

1. When I log in, my token reflects the claimscollection.read and claimscollection.write scopes.
2. The token claims as per each service rule, will not allow me to create claims on the /claims endpoint.

4.2 Scopes

In general, scopes should reflect and communicate transparently the owner's intent at the highest level, as to what kind of access to the endpoint she is consenting an application to have.
The scopes described here are the least common denominator for scopes that requesting applications can ask to receive through a participating bank's authorization server, directly in code in the appropriate use cases, or with the resource owner potentially involved in others. If the latter, the client implementation must expect the final granted scopes to possibly reflect a subset of those originally requested.
The authorization mechanism in each bank will, of course, further define access based on internal rules, e.g. their specific product offerings or service agreements.

Table 4.1: General overview of available scopes.

Scope	Description
payments	Payment scope without prefix, when specific tokens are not issued for individual payments
pis:{PaymentId}	Prefix for payment scope, when dynamic scopes are supported by provider
accounts	Account scope without prefix, when user access is not specified by the optional consent endpoint
ais:{ConsentId}	Account consent scope
claims	Claim scope
claimscollection	Collection Claims scope
documents	Document scope
consents	Consent scope
openid	Standard Open Id Scope
offline_access	Oauth scope to request use of refresh tokens

4.2.1 Payment endpoints

Depending on the implementation, the endpoints for payments can require either a general payments scope, as for the root resource, or dynamic scopes that dynamically link this particular request to a known context. In the latter case, the NextGenAPI *pis* pattern is used for overall compatibility.

Table 4.2: Payments and possible scopes.

Payments EndPoint	Scope
/v1/{payment-service}/{payment-product}:	payments
/v1/{payment-service}/{payment-product}/{paymentId}:	payments, pis:{paymentId}
/v1/{payment-service}/{payment-product}/info/{Query-X-Request-ID}:	payments, pis:{paymentId}
/v1/{payment-service}/{payment-product}/{paymentId}/status:	payments, pis:{paymentId}
/v1/{payment-service}/{payment-product}/{paymentId}/authorisations:	payments, pis:{paymentId}
/v1/{payment-service}/{payment-product}/{paymentId}/authorisations/{authorisationId}:	payments, pis:{paymentId}
/v1/{payment-service}/{payment-product}/{paymentId}/cancellation-authorisations:	payments, pis:{paymentId}
/v1/{payment-service}/{payment-product}/{paymentId}/cancellation-authorisations/{authorisationId}:	payments, pis:{paymentId}

4.2.2 Accounts endpoints

Depending on the implementation, the endpoints for accounts can require either a general accounts scope, as for the root resource, or dynamic scopes that dynamically link this particular request to a known consent. In the latter case, the NextGenAPI *ais* pattern is used for overall compatibility.

Table 4.3: Accounts and possible scopes.

Accounts EndPoint	Scope
/v1/accounts:	accounts, ais:{consentId}
/v1/accounts/{account-id}:	accounts, ais:{consentId}
/v1/accounts/{account-id}/balances:	accounts, ais:{consentId}
/v1/accounts/{account-id}/transactions:	accounts, ais:{consentId}

Accounts EndPoint	Scope
/v1/accounts/{account-id}/transactions/{transactionId}:	accounts, ais:{consentId}

4.2.3 Card endpoints

Depending on the implementation, the endpoints for accounts can require either a general accounts scope, as for the root resource, or dynamic scopes that dynamically link this particular request to a known consent. In the latter case, the NextGenAPI *ais* pattern is used for overall compatibility.

Table 4.4: Card accounts and scopes.

Card EndPoints	Scope
/v1/card-accounts:	accounts, ais:{consentId}
/v1/card-accounts/{account-id}:	accounts, ais:{consentId}
/v1/card-accounts/{account-id}/balances:	accounts, ais:{consentId}
/v1/card-accounts/{account-id}/transactions:	accounts, ais:{consentId}

4.2.4 Currency endpoints

Currencies are an example of an open data endpoint that does not require a particular scope, and only included here for completeness to make that clear.

Table 4.5: Currency endpoints.

Currency EndPoint	Scope
/v1/currencies:	NA
/v1/currencies/sources:	NA
/v1/currencies/{base-currency}/rates:	NA
/v1/currencies/{quote-currency}/rates/{base-currency}:	NA
/v1/currencies/{quote-currency}/rates/{base-currency}/history:	NA

4.2.5 Documents endpoints

For endpoints related to documents, two scopes are possible for read or write.

Table 4.6: Required document scopes.

Documents EndPoint	Scope
/v1/documents/{document-store-location}/{sender-kennitala}/{documents-id}:	documents.read, documents.write
/v1/documents/{documentStoreLocation}:	documents.read, documents.write
/v1/documents/{documentStoreLocation}/types:	documents.read

4.2.6 Consents endpoints

For consents, scopes can specify either read or write.

Table 4.7: Required consents scopes.

Consents EndPoint	Scope
/v1/consents/	consents.read, consents.write

4.2.7 Claim template endpoints

Claim templates can only be queried, so the scope is read-only.

Table 4.8: Required claim template scopes.

Claim Templates EndPoint	Scope
/v1/claimtemplates	claims.read
/v1/claimtemplates/{claimTemplateId}	claims.read

4.2.8 Claim endpoints

For endpoints related to claim resources, the users can either be primary claimants or in the role of a secondary collection agent. It is not expected that service providers support combining these two roles, but all authorization servers should accept either as appropriate per endpoint. Additional access restrictions and business logic can of course apply as indicated by documentation provided by the service provider.

Table 4.9: Required claim scopes.

Claims EndPoint	Scope
/v1/claimtemplates	claims.read

Claims EndPoint	Scope
/v1/claimtemplates/{claimTemplateId}	claims.read
/v1/claims/{claimId}	claims.read, claims.write, claimscollection.read, claimscollection.write
/v1/claims/{claimId}/transactions	claims.read, claimscollection.read
/v1/claims/{claimId}/history	claims.read, claimscollection.read
/v1/claims	claims.read, claims.write, claimscollection.read
/v1/batches	claims.write, claimscollection.write
/v1/batches/{batchId}	claims.read, claimscollection.read
/v1/claims/transactions	claims.read, claimscollection.read

5 Idempotency

Certain services of IOBWS and selected methods of those services shall support idempotency to prevent duplication of actions with side effects. This is especially important in the case of initiating single payments, payment batches and claim batches but may apply in other cases.

To use idempotency, insert an idempotent key into the header of the POST request to the APIs.

5.1 Functionality

An idempotency key is a unique value generated by the client that the resource service uses to recognize subsequent retries of the same request.

Idempotent keys MUST be guaranteed to be recognized by the service implementer for at least 24 hours but MAY be expired after that.

The idempotent key MUST be a valid UUID.

The uniqueness of the message is only based on the idempotent key and the request payload is ignored.

5.2 Error Scenarios

When an operation with an idempotent key is received, and the idempotent key is in storage, the following will happen:

- If the previous operation succeeded, the same result from the previous payment is returned.
- If the previous operation does not manage to complete, the API will return with status code 409 (Conflict). The API will not try to execute the operation again.
- If the previous operation failed, for whatever reason the error code from the previous payment is returned. Example: 400 Bad Request/500 Server Error

Optional:

If the previous operation has not yet been completed, the API will wait for a moment and give the previous operation a chance to complete.

If the previous operation manages to complete during that time, the same result from the previous payment is returned.

5.3 Best Practices

It´s recommended to use the same idempotent key again when the client system does not receive a response from the API, for example, HttpStatusCode:

- 408 Request Timeout
- 409 Conflict
- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout

An example of a header submitted with the idempotent key element:

Listing 5.1: Example of a http header with the *Idempotent-Key* data element.

```
    curl -X POST "/v1/payments/sepa-credit-transfers" -H "accept: text/plain"  -H "Idempotent
        -Key:uuid" -H "xRequestID: <random-UUID>" -H "pSUIPAddress: <IP-Address>" -H "
        Authorization: Bearer <Access-Token>" -H "Content-Type: application/json-patch+json"
        -d "<payment-payload>"

    POST /v1/payments/sepa-credit-transfers
    Host:
    Content-Type: application/json-patch+json
    Authorization: Bearer oauth2_token
    Idempotent-Key: 123e4567-e89b-12d3-a456-336655440000
```

6 Merge Patching

RFC 7396 [3] defines a simple format for describing modifications to a JSON document. It is primarily designed for use with the HTTP PATCH method and is especially suited to JSON documents that are structured as objects. The RFC should be considered as a reference for how patching is used in the API specifications that are part of the version 3.x series of the Icelandic Online Banking Services (IOBWS) APIs. Some of the key points in how patching should be applied focus on describing changes by “merging” a patch document with the target JSON document, rather than listing a series of operations:

- Addition/Replacement: If a key in the patch does not exist in the target, it is added. If it exists, its value is replaced with the one from the patch.
- Deletion: If a key’s value in the patch is null, the key is removed from the target.
- Recursive Merging: If a key’s value is an object, the patch is applied recursively to merge with the target’s object value.

  Further instructions, explanations and examples can be found in the RFC document.

Bibliography

[1] ISO 20022. *Financial services - universal financial industry message scheme.*

[2] OpenAPI v3.0.1. *The OpenAPI Specification (OAS) by the OpenAPI Initiative, a Linux Foundation Collaborative Project.*

[3] RFC 7396 - JSON Merge Patch. *RFC 7396*.

[4] NextGenPSD2 v1.3.8. *The Berlin Group NextGenPSD2 Access to Account Framework*.

[5] Lög 55/2019. *Lög um rafræna auðkenningu og traustþjónustu fyrir rafræn viðskipti*.

[6] Semver 2.0.0. *Semantic Versioning Specification*.

[7] EU 910/2014. *Regulation of the European Parliament and of the Council of on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC*.