CEN

WORKSHOP

AGREEMENT

CWA 17852

February 2022

ICS 35.240.40

English version

Extensions for Financial Services (XFS) - XFS4IoT Specification - Release 2021-1 Release Candidate

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its Members.

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

Ref. No.:CWA 17852:2022 E

Table of Contents

Foreword

This CEN Workshop Agreement (CWA 17852:2022)has been developed in accordance with the CEN-CENELEC Guide 29 “CEN/CENELEC Workshop Agreements – A rapid prototyping to standardization” and with the relevant provisions of CEN/CENELEC Internal Regulations - Part 2. It was approved by a Workshop of representatives of interested parties on 2022-01-10, the constitution of which was supported by CEN following several public call for participation, the first of which was made on 1998-06-24. However, this CEN Workshop Agreement does not necessarily include all relevant stakeholders.

The final text of this CEN Workshop Agreement was provided to CEN for publication on 2021-01-12.

The following organizations and individuals developed and approved this CEN Workshop Agreement:

- ATM Japan LTD
- AURIGA SPA
- BANK OF AMERICA
- CASHWAY TECHNOLOGY
- CHINAL ECTRONIC FINANCIAL EQUIPMENT SYSTEM CO.
- CIMA SPA
- CLEAR2PAY SCOTLAND LIMITED
- DIEBOLD NIXDORF
- EASTERN COMMUNICATIONS CO. LTD – EASTCOM
- FINANZ INFORMATIK
- FUJITSU FRONTECH LIMITED
- FUJITSU TECHNOLOGY
- GLORY LTD
- GRG BANKING EQUIPMENT HK CO LTD
- HESS CASH SYSTEMS GMBH & CO. KG
- HITACHI OMRON TS CORP.
- HYOSUNG TNS INC
- JIANGSU GUOGUANG ELECTRONIC INFORMATION TECHNOLOGY
- KAL
- KEBA AG
- NCR FSG
- NEC CORPORATION
- OKI ELECTRIC INDUSTRY SHENZHEN
- OKI ELECTRONIC INDUSTRY CO

- PERTO S/A
- REINER GMBH & CO KG
- SALZBURGER BANKEN SOFTWARE
- SIGMA SPA
- TEB
- ZIJIN FULCRUM TECHNOLOGY CO

Attention is drawn to the possibility that some elements of this document may be subject to patent rights. CEN-CENELEC policy on patent rights is described in CEN-CENELEC Guide 8 “Guidelines for Implementation of the Common IPR Policy on Patent”. CEN shall not be held responsible for identifying any or all such patent rights.

Although the Workshop parties have made every effort to ensure the reliability and accuracy of technical and non-technical descriptions, the Workshop is not able to guarantee, explicitly or implicitly, the correctness of this document. Anyone who applies this CEN Workshop Agreement shall be aware that neither the Workshop, nor CEN, can be held liable for damages or losses of any kind whatsoever. The use of this CEN Workshop Agreement does not relieve users of their responsibility for their own actions, and they apply this document at their own risk. The CEN Workshop Agreement should not be construed as legal advice authoritatively endorsed by CEN/CENELEC.

1. Scope

XFS4IoT has been identified as a successor to XFS 3.x to meet the following requirements:

1. Replace the XFS and J/XFS standards in the marketplace.
2. Target industries – Retail Banking.
3. Operating System Agnostic and Technology and Language Adaptable.
4. Multi-Vendor – Able to run common core high level functionality on multiple vendors hardware, while providing access to finer level device API granularity.
5. Flexibility – enabling new hardware topologies, device types and functionality to be rapidly adapted.
6. Support end to end application level security.
7. Should not prevent the use of a low resource computing environment.
8. Provide a good developer experience by providing a well-documented API that is easy to learn, is quick to market and reduces risk by exposing an unambiguous interface.
9. Leverage existing standards.

Within the overall requirements specified in the Charter, the opportunity has been taken to solve some of the issues with the 3.x interface while retaining all the same functionality:

1. Binary data structures makes adding new functionality difficult due to compatibility issues, leading to multiple redundant versions of the same command appearing in many of the existing device classes. To resolve this, a flexible text based approach has been adopted including the wide use of default parameters.
2. Compound devices have been difficult for applications to implement, particularly cash recycling. Addition of other shared functionality such as end to end security would make the use of compound devices more prevalent. Compound devices are removed in XFS4IoT, a single Service can support as many interfaces as required to support its requirements.

Migration from and to 3.x is a major consideration to support adoption of XFS4IoT. While a lot of duplication has been removed (for example the Card Reader interface has fewer commands and events defined than the equivalent 3.x IDC specification), all the same IDC commands and events can be implemented. In some cases, this is achieved by having shared common commands such as Common.Status which replaces all the 3.x WFS_INF_XXX_STATUS commands.

2. API

This chapter defines the API functionality and messages. It defines the XFS4IoT API including but not limited to:

- System Architecture
- Message Definition
- End to End Security

XFS4IoT defines a system consisting of Services provided by one or more vendors. Each Service can support one or more interfaces as required to meet the requirements of the device or function it supports, so for example a Cash Recycling device will need the following interfaces to supply all the device's functionality:

- Common, which defines functionality common to all devices
- CashManagement, which defines functionality common to all cash handling devices
- CashAcceptor, which defines functionality common to all cash accepting devices
- CashDispenser, which defines functionality common to all cash dispensing devices
- Storage, which defines functionality common to devices which store items

Additional interfaces can be added as required for example KeyManagement to support encryption key management.

The following sections describe how clients and services create connections and send messages to each other.

2.1 References

ID	Description
api-1	JSON (https://www.json.org/)
api-2	XFS Interface Specification, End to End (E2E) for XFS/XFS4IoT Programmer's Reference
api-3	WebSockets - IETF RFC 6455

2.2 WebSockets Connections

Multiple services can be supplied by multiple vendors. This standard doesn't require coordination between these different vendors, or between the service publishers and the service client. It is possible to operate a system with components from multiple hardware vendors, and with third party applications, without the prior knowledge of any party.

This specification covers an environment using WebSockets (ref. api-3) to communicate between services and applications, either on a single machine or across a network.

This section covers both the process for publishing a service such that it can be discovered, and the discovery process used by the service client.

There is also a clear definition of responsibility for each component in the system, including when there are dependencies between components. There are no shared components required to coordinate the system.

The underlying network can use any protocol that supports WebSockets such as IPv4 or IPv6. Nothing in this document requires any particular underlying protocol.

2.2.1 Overview

In this standard there are two types of "endpoint"; publisher and service. Each endpoint, of either type, is published by a single software/hardware vendor. A publisher endpoint is used for service discovery, to discover service endpoints. A single service endpoint can expose multiple "services", where each service typically represents a single piece of hardware. A single machine (or a single IP address) may expose multiple publisher and service endpoints from different vendors. A "client" application may consume multiple services from multiple service endpoints on the same machine, or across multiple machines.

On startup of the machine, any software services attempt to claim access to individual network ports using the underlying operating system mechanism. Ports are claimed sequentially from a known sequence. Each port becomes an endpoint that can publish multiple services from a single vendor.