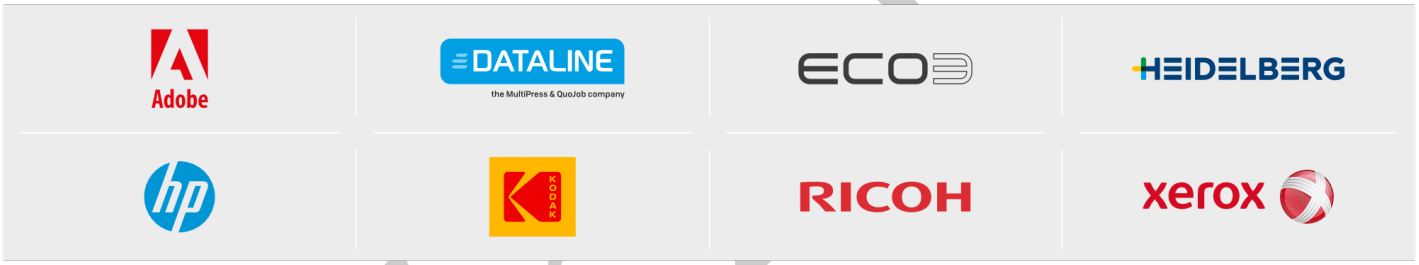
CIP4 THANKS ITS PARTNER LEVEL MEMBERS



Contents

Chapter 1

Introduction

This document defines the technical specification for the Exchange Job Definition Format (XJDF) and its counterpart, the Exchange Job Messaging Format (XJMF).

XJDF is a technology that allows systems from many different vendors to interoperate in automated workflows. While technically it is an XML software specification, it is more importantly a means to connect multiple vendor solutions to a workflow solution for automation.

XJDF 2.0 was the first major version update of JDF. It is such a major update that we decided to provide a new name for the XML root element: XJDF. Whereas the minor revisions were at least nominally backwards compatible, XJDF is a major redesign that takes more than a decade of experience into account. XJDF 2.1 is a minor update and is backwards compatible with XJDF 2.0.

Note: The specification uses two forms for references to XJDF/XJMF (the general concept of the specification) and XJDF/ XJMF (for specific reference to the root element of an XML instance).

This document is intended for use by programmers and systems integrators. It provides both the syntactical requirements for the elements and attributes of XJDF and XJMF as well as requirements for devices and controllers to act upon the data. In this first chapter, we present the concept of XJDF, and its relationship to JDF and other industry standards.

1.1 Further Information

Additional information such as application notes and examples can be found on the CIP4 website at http://www.CIP4.org and the CIP4 technical website at https://confluence.cip4.org.

1.1.1 NMTOKEN repository

Open lists are marked with a data type of NMTOKEN or NMTOKENS and contain a list of suggested values. The list of values may be incomplete and sometimes needs to be extended with new values without updating the specification, e.g. when a new domain ICS is developed.

Additional, suggested values are maintained in the CIP4 technical discussion area at https://confluence.cip4.org. In order to avoid different extension values being used for the same purpose, vendors are encouraged to check this area prior to using new values. In the event that no existing extension exists then vendors are further encourged to submit their extensions to CIP4 using the CIP4 issue tracking system at https://jira.cip4.org.

1.1.2 Errata

Although great care has been taken to ensure that this specification is correct and complete, some errors cannot be avoided. CIP4 therefore maintains an online errata repository in its technical discussion area at https://confluence.cip4.org. A copy of the original specification with annotations identifying the errata is also published and can be found at https://confluence.cip4.org.

The corrections in the errata override the published specification.

1.2 Background on XJDF

XJDF is an extensible, XML-based data interchange format built upon more than twenty years of experience with JDF.

XJDF is an interchange data format that can be used by a system of controllers, devices and MIS, which together produce printed products. It provides the means to describe print jobs in terms of the products eventually to be created, as well as in terms of the work steps needed to create those products. XJDF provides a syntax to explicitly specify the details of processes, which might be specific to the devices that execute the processes.

XJDF is aligned with a communication format known as the Exchange Job Messaging Format or XJMF. XJMF provides the means for production components of an XJDF workflow to communicate with controllers such as MIS. It gives MIS and other controllers the ability to receive information from devices or other controllers about the status of jobs and devices. XJDF and XJMF are maintained and developed by CIP4 (http://www.CIP4.org).

1.3 Design Criteria for XJDF

The major conceptual change is that XJDF no longer attempts to model the entire job as one large "job ticket" but rather specifies an interchange format between 2 applications that are assumed to have an internal data model that is not necessarily based on XJDF. Thus each XJDF ticket specifies a single transaction between two parties. A single job may be modeled as one or more XJDF transactions.

The following criteria were taken into account in this redesign:

- XJDF should be simple to use.
- The number of methods to describe similar traits should be as limited as possible, ideally one.
- XJDF should be compatible with the latest XML tools to simplify development.
- Simple XPath expression to reference XJDF traits.
- Direct use of ID-IDREF pairs for referencing distributed data within an XML document.
- Use of XML schema rather than proprietary data structures to describe device capabilities.
- The semantics of JDF 1.x should be retained and mapping between JDF 1.x and XJDF should be simple.
- Change orders (Modifications of submitted jobs) should be easy to describe.

These requirements lead to some significant modifications that are not syntactically backwards compatible, but can easily be converted using JDF 1.x aware middleware.

1.3.1 Simplify and reduce variations

JDF 1.x allowed shorthand for some simple cases. What seemed reasonable actually made things more complex, since both shorthand and the long version had to be implemented. For instance, amount related attributes could be found either directly in a ResourceLink or in a ResourceLink/AmountPool/PartAmount. XJDF removes much of this variability.

1.3.1.1 Reduce the barrier of entry

Simple tasks should be easy to describe. In such cases the XJDF should be capable of being described as a short list of simple XPaths.

1.3.1.2 Single XJDF

JDF 1.x allowed for multiple 'JDF' nodes within one ticket. This grouping of multiple nodes in process groups resulted in many variations of JDF for the same or similar requirements. Version 2.x has exactly one XJDF element, namely the root element; this contains no XJDF child nodes. This means there can be no ambiguity about where to locate and retrieve a given trait.

1.3.1.3 Replace abstract data types with explicit elements and children

Abstract elements are more concise to write, but inherited traits also tend to be overlooked by newcomers to a specification. If elements are designed to be final with sub-elements, each specification entry can be found by searching for the explicit element name.

1.3.1.4 Remove ResourceLinks

XJDF allowed specification of interdependencies of processes using 'ResourceLink' elements. In most cases, this feature is not required if the controller maintains an internal job model. Therefore XJDF does not provide mechanisms to describe process networks within a single XJDF.

The Process / Resource model has been conceptually retained. But since there is only one XJDF element per XJDF transaction, reuse of resources is no longer an issue and 'ResourceLink' elments have been merged with their respective resources. Thus data that belongs together is also stored in the same region of the XML.

Chapter 2

Overview

TBD

Chapter 3

Structure

A single XJDF describes the information about a job or process step that is transferred from a controller to a device. The scope of the exchanged information varies depending on the nature of the recipient device. An XJDF that is targeted at an individual device will typically contain only the details that are required by that device, along with some optional information about the final product. Multiple work steps belonging to one job that need to be submitted from a controller to a workflow system that controls multiple devices SHALL be submitted as a separate XJDF for each work step. These MAY be packaged together and submitted as one or more transactions. See Chapter 9 Building a System for details of packaging and referencing of the individual XJDF.

3.1 XJDF

The top-level element of an XJDF instance SHALL be an XJDF element. See Table 3.1 XJDF below for details. XJDF elements MAY be embedded within other XML documents.

XJDF/@Types defines whether an XJDF specifies an end product or a list of processes that SHALL be executed. XJDF that are created by print buyers typically describe only the desired product rather than manufacturing process details. XJDF that describe finished products SHALL have a value of XJDF/@Types that contains "Product". If additional process information that is not defined in the ProductList is required, this information SHOULD be provided in ResourceSet elements.

ProductList MAY be provided in a process XJDF for informational purposes.

schema ?	URL	schema SHOULD reference the JSON schema for XJDF
Category ?	NMTOKEN	Category specifies the named category of this XJDF. Controllers SHOULD specify @Category for processes that have many optional values in @Types. This allows processors to identify the general purpose of an XJDF without parsing the @Types field. For instance, a RIP for final output and a RIP for proof process have identical @Types attribute values, but have @Category = "RIPing" or @Category = "ProofRIPing", respectively. **Values include those from** Node Categories **Note**@Category MAY also be the name of a Gray Box defined by an ICS document. See Section 1.9.2 Interoperability Conformance Specifications for details.
CommentURL ?	URL	CommentURL SHALL refer to an external, human-readable description of this XJDF.

Table 3.1: XJDF