Open Standard Job Ticket API (JTAPI) Specification
Version *working draft*

FSG Job Ticket Working Group
Maintained by:
Claudia Alimpich
303-924-4418
alimpich@us.ibm.com
August 7, 2002

1. Deliverables

1) This Open Source Job Ticket API (JTAPI) Specification that defines and describes the JTAPI, the scope of the project, the content of each version.
2) Job Ticket API UML diagrams that show the objects, their content, and their relationships.
3) Implementation of the JTAPI? Implemented in what programming languages?

2. Capabilities and limitations of the JTAPI

1) The first phase of the JTAPI, it will provide the ability to specify how a job is to be printed, i.e. a description of the job or a job's properties/attributes.

2) The first phase of the JTAPI will not allow the job ticket to be updated to contain job status or error information.

3) The intention is that the JTAPI not deal with security, authentication, or log in information. It is left up to the transport to handle these things. The job ticket can contain such things as the job submitter's name, but nothing in the job ticket is secure.

4) The subjects of audit, logging, and accounting will not be included in the first phase of the JTAPI. They will be considered for inclusion in future phases.

5) Since the JTAPI is implementation independent, the resulting job ticket may not even be XML, so it is not clear whether the JTAPI should ever concern itself with XML Schema. However, a JTAPI implementation can and may.

6) The JTAPI will provide support for utf-8 encoding. There is an open source library called "International Components for Unicode" and is available at http://oss.software.ibm.com/icu/docs.html that provides unicode support including conversion utilities.

7) The enumerations that are not likely to be extended, will be integer-based, and those that are likely to be extended will be strings. The general feeling is that most of the enumerations will be integer-based, but there is a concern about how it will match up with the Capability API. Also, will it be possible to have both string and integer based enumerations in the implementation and not be confusing. More discussion on this to come.

8) There are operations that can be performed at all levels, page-range, document, and the job (e.g. finishing, media, output bin selection). How they should be handled when more than one is specified needs to be defined.

9) The JTAPI must support vendor-specific extensions.

10) Compressing or the handling of compressed job tickets will not be included as part of the first version of the JTAPI.

11) Decisions made about the charset/encoding of the JTAPI.
    - The JTAPI will support the specification of the charset/encoding of the JTAPI.
    - The JobTicketInfo object's data member apiCharset is a string that the implementation will initialize to be the same as the charset of the environment, if it can be determined. If it cannot be determined then apiCharset will be initialized to utf-8.
    - The application can use the generic set method/function to explicitly set the value of the jt-api-charset attribute which will set the apiCharset data member and change the charset of the JTAPI.
    - It is up to the implementation to decide which and how many charsets it will support for the JTAPI. At a minimum an implementation of the JTAPI must support utf-8.

12) Decisions made about the charset/encoding of the job ticket itself.
    - The JTAPI will support the specification (as a parameter) of the charset/encoding of the job ticket when a job ticket is read.
    - An implementation of the JTAPI must be able to determine the charset of a job ticket that it read in, as follows:
        a. As specified in the job ticket (e.g. for an XML-based job ticket the "encoding" declaration).
        b. Externally (e.g. the charset declaration on the URL).
        c. Some other way. e.g. the charset could be hardcoded by the implementation.

      If the charset that was provided as a parameter when the job ticket was read in is not the same as the charset of the job ticket as determined by the implementation, the implementation can treat this as an error condition or can ignore the provided charset parameter and continue.
    - The JobTicketInfo object's jtCharset data member is a string that defines the charset of an existing job ticket that was read by the JTAPI. jtCharset is initialized to the charset as determined by the implementation (as defined above).
    -

- The application can use the generic get method/function to retrieve the value of the jt-charset attribute which will get the value of the jtCharset data member.
- When writing a job ticket the jtCharset data member is overridden by the provided charset parameter. So the job ticket will be written in the charset as specified by the charset parameter.

13) When comparing and interpreting character sets, the following rules will apply:
    - case will be ignored (e.g. UTF-8 and utf-8 are the same)
    - punctuation that is not alphanumeric will be ignored (e.g. utf8, utf.8, and utf-8 are the same)
    - spaces will be ignored (e.g. utf 8 and utf-8 are the same).

14) Implementations of the JTAPI for imbedded systems will be allowed to only implement a subset of the JTAPI.

15) Decisions made about the type of the job ticket.
    - An implementation may support more than one type of job ticket (e.g JDF and PWG).
    - The type of the job ticket can be provided as a parameter when a job ticket is read in or the implementation can be told to try and detect the job ticket's type. If the type that is provided as a parameter is not consistent with the implementation's determination of the job ticket type, then the implementation can treat this as an error condition or can ignore the provided type parameter and continue.
    - The JobTicketInfo object's type data member is initialized to the type of the job ticket as determined by implementation. For a new job ticket since there is not a job ticket to read, the type data member is set to the value of the provided type parameter unless the implementation is told to detect the type in which case it is up to the implementation to initialize the type data member.
    - The application can use the generic get/set method/function to retrieve the value of the jt-type attribute which will get/set the value of the type data member.
    - When writing a job ticket the type data member is overridden by the provided type parameter. So the job ticket will be written as the type as specified by the type parameter. If the implementation is told to try and determine the job ticket's type, then the implementation will determine what type of job ticket to write, which may be the type as specified by the type data member.

3. Future additions to this specification

The following are not yet included in this specification, but may be considered in the future.

1) If, how, what audit, logging, and accounting information should be supported in the job ticket and by the JTAPI.