-OSTA.org-
Optical Storage
Technology Association

Digital Music, Photo, and Video Collections

MPV Internet Profile Specification

Revision 0.40
Working Draft

8 December 2003

IMPORTANT NOTICE

This document is a working draft for review by OSTA members and approved parties. It is a draft document and will be updated, replaced, or obsoleted by other documents at any time and without notice. It is inappropriate to use OSTA Working Draft documents as reference materials, to cite them in other publications, or to refer to them as anything other than a "work in progress".

NOT FOR DISBTRIBUTION ON A PUBLIC WEBSITE

This document is available at http://www.osta.org/mpv/mpvmbrs/specs/MPVInternet-Spec-0.40WD.PDF

POINTS OF CONTACT

OSTA	*Technical Content*
David Bunzel OSTA President Tel: +1 (408) 253-3695 Email: dbunzel@osta.org http://www.osta.org *MPV Website* http://www.osta.org/mpv/	Pieter van Zee MPV Working Group Chairman Editor, MPV Specification Tel: +1 541-715-8658 Email: Pieter_van_Zee@hp.com Felix Nemirovsky Chairman, MultiRead Subcommittee Tel: +1 415 643 0944 Email: felixn@pacbell.net

ABSTRACT

The MPV Internet Profile Specification defines the MPV implementation practices and schema for using MPV manifests to interact between a client and server connected using internet HTTP protocols.

COPYRIGHT NOTICE

REVISION HISTORY

Revision	Date	Comments
0.40	8 December 2003	First draft, but based on specification work previously developed by Hewlett-Packard Company.

LICENSING
IMPORTANT NOTICES

(a) THIS DOCUMENT IS AN AUTHORIZED AND APPROVED PUBLICATION OF THE OPTICAL STORAGE TECHNOLOGY ASSOCIATION (OSTA). THE SPECIFICATIONS CONTAINED HEREIN ARE THE EXCLUSIVE PROPERTY OF OSTA BUT MAY BE REFERRED TO AND UTILIZED BY THE GENERAL PUBLIC FOR ANY LEGITIMATE PURPOSE, PARTICULARLY IN THE DESIGN AND DEVELOPMENT OF WRITABLE OPTICAL SYSTEMS AND SUBSYSTEMS. THIS DOCUMENT MAY BE COPIED IN WHOLE OR IN PART PROVIDED THAT NO REVISIONS, ALTERATIONS, OR CHANGES OF ANY KIND ARE MADE TO THE MATERIALS CONTAINED HEREIN.

(b) COMPLIANCE WITH THIS DOCUMENT MAY REQUIRE USE OF ONE OR MORE FEATURES COVERED BY THE PATENT RIGHTS OF AN OSTA MEMBER, ASSOCIATE OR THIRD PARTY. NO POSITION IS TAKEN BY OSTA WITH RESPECT TO THE VALIDITY OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT, WHETHER OWNED BY A MEMBER OR ASSOCIATE OF OSTA OR OTHERWISE. OSTA HEREBY EXPRESSLY DISCLAIMS ANY LIABILITY FOR INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF OTHERS BY VIRTUE OF THIS OSTA DOCUMENT, NOR DOES OSTA UNDERTAKE A DUTY TO ADVISE USERS OR POTENTIAL USERS OF OSTA DOCUMENTS OF SUCH NOTICES OR ALLEGATIONS. OSTA HEREBY EXPRESSLY ADVISES ALL USERS OR POTENTIAL USERS OF THIS DOCUMENT TO INVESTIGATE AND ANALYZE ANY POTENTIAL INFRINGEMENT SITUATION, SEEK THE ADVICE OF INTELLECTUAL PROPERTY COUNSEL AND, IF INDICATED, OBTAIN A LICENSE UNDER ANY APPLICABLE INTELLECTUAL PROPERTY RIGHT OR TAKE THE NECESSARY STEPS TO AVOID INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT. OSTA EXPRESSLY DISCLAIMS ANY INTENT TO PROMOTE INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT BY VIRTUE OF THE EVOLUTION, ADOPTION, OR PUBLICATION OF THIS OSTA DOCUMENT.

(c) ONE OR MORE PATENT HOLDERS HAVE FILED STATEMENTS OF WILLINGNESS TO GRANT A LICENSE, ON REASONABLE AND NONDISCRIMINATORY TERMS, ON A RECIPROCAL BASIS, UNDER PATENT CLAIMS ESSENTIAL TO IMPLEMENT THIS SPECIFICATION. FURTHER INFORMATION MAY BE OBTAINED FROM OSTA.

(d) OSTA MAKES NO REPRESENTATION OR WARRANTY REGARDING ANY SPECIFICATION, AND ANY COMPANY USING A SPECIFICATION SHALL DO SO AT ITS SOLE RISK, INCLUDING SPECIFICALLY THE RISKS THAT A PRODUCT DEVELOPED WILL NOT BE COMPATIBLE WITH ANY OTHER PRODUCT OR THAT ANY PARTICULAR PERFORMANCE WILL NOT BE ACHIEVED. OSTA SHALL NOT BE LIABLE FOR ANY EXEMPLARY, INCIDENTAL, PROXIMATE OR CONSEQUENTIAL DAMAGES OR EXPENSES ARISING FROM THE USE OR IMPLEMENTATION OF THIS DOCUMENT. THIS DOCUMENT DEFINES ONLY ONE APPROACH TO COMPATIBILITY, AND OTHER APPROACHES MAY BE AVAILABLE IN THE INDUSTRY.

(e) THIS DOCUMENT IS A SPECIFICATION ADOPTED BY OSTA. THIS DOCUMENT MAY BE REVISED BY OSTA AT ANY TIME AND WITHOUT NOTICE AND USERS ARE ADVISED TO OBTAIN THE LATEST VERSION. IT IS INTENDED SOLELY AS A GUIDE FOR ORGANIZATIONS INTERESTED IN DEVELOPING PRODUCTS WHICH CAN BE COMPATIBLE WITH OTHER PRODUCTS DEVELOPED USING THIS DOCUMENT. THIS DOCUMENT IS PROVIDED “AS IS”.

(f) THE MPV NAME, THE MPV LOGO, AND THE MusicPhotoVideo NAME ARE TRADEMARKS OF OPTICAL STORAGE TECHNOLOGY ASSOCIATION, INC. ALL OTHER TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. THESE OSTA TRADEMARKS MAY NOT BE USED EXCEPT FOR JOURNALISTIC PURPOSES WITHOUT AN EXPLICIT LICENSE FROM OSTA.

Contents

Table of MPV Internet Profile Requirements

The following table is a comprehensive list of the requirements specified by the MPV Internet Profile specification, ordered by Requirement Number. This list can provide a quick reference guide to the specification and also is useful to quickly lookup more information about a specific requirement by number.

WRITER REQUIREMENTS

Error! No table of figures entries found.

READER REQUIREMENTS

Error! No table of figures entries found.

COMMON REQUIREMENTS

Error! No table of figures entries found.

Chapter 1: Introduction

1.1 Introduction to MPV

MPV (MusicPhotoVideo) is an open, multiplatform specification for playlists and asset management of digital music, photo, and video files. MPV makes easier the representation, exchange, processing and playback of collections of digital media content, including music, still images, stills with audio, still sequences, video clips, and audio clips.

The MPV Internet Profile Specification, which is defined by this specification, defines the MPV implementation practices and schema for using MPV-compliant content to interact between a client and server connected using internet HTTP protocols.

MPV playlists and albums can be enjoyed in consumer electronics products such as a CD or DVD player or on a PC. MPV playlists and the content they reference are unlike audio CDs and DVD-Video discs because they store the multimedia content like MP3 and JPEG files in formats used by PCs on a "data CD".

MPV is an open format developed under the leadership of the Optical Storage Technology Association (OSTA) and available from OSTA at no cost. Information regarding the MPV Specification, SDK for software developers, and logo licensing program can be found at www.osta.org/mpv.

The MPV Logo and sublogos provide a consumer-oriented mark of compatibility that can be used by creative applications and devices, on storage media, and by playback devices and applications. Use of the MPV Logo and sublogos is governed by a licensing program that defines the requirements and costs.

SITUATION TODAY

Consumers create CDs full of personal digital content – music, photo, and video content in PC file formats like MP3, JPEG and MPEG – captured by digital cameras, photofinishing retailers, personal music collections, and PC multimedia applications. The expectation and desire of consumers is to enjoy the playback experience not only in PCs but wherever a player can go -- in their home entertainment environment, in their car or in their pocket.

Today, most consumer electronic (CE) devices do not recognize the content on CDs created by PC applications or retail services or do so poorly. Because each application stores the content on a disc uniquely, there is no standard way for CD and DVD players to recognize and playback the content. And the user playback experience is different between each CE device.

Without a standard method for organization and access, the CE device can take many minutes reading through large collections of multimedia content or will present filesystem views of the data. Consumers get frustrated with trying to find and quickly access their desired music, photo, or video content.

CE devices are starting to add support for PC formats. MP3 for music has enjoyed wide spread adoption in DVD player, car stereo, and personal music systems. Support for JPEG for photos is growing in consumer electronics products. MPEG1 and MPEG2 video is already broadly adopted by both PC and CE industries. Additional formats are emerging and growing in popularity, such as Windows Media Audio (WMA), ATRAC3, MPEG4 Audio (AAC), and more.

MPV BENEFITS CONSUMERS

When applications and application both write and read MPV, consumers will enjoy enhanced interoperability of content between applications and devices from any vendor. This gives consumers more choice and vendors greater ability to innovate and differentiate.

1.2 MPV Profiles

The MPV specification is developed in a modular manner and in phases. This results in "profiles". Each profile in MPV defines only those formats and practices that are necessary for the key tasks targeted by the profile. MPV Internet Profile makes use of the Core specification, which defines the overall framework of all MPV profiles.

1.3 MPV Internet Profile Introduction

The MPV Internet Profile is an interface to internet services that supports interaction using commonly available web browsers and established protocols and browser-specific capabilities.

A principle objective of the MPV Internet Profile is to make internet services readily accessible to the broad installed base of users who have access to common web browsers. Generally speaking, any internet service can provide a MPV Internet Profile interface by adding support for the required capabilities and interfaces of this document. In this manner, the functionality of a given service can be extended to support interaction via the common web browser. This is likely most meaningful for consumer-to-business services.

In the MPV Internet Profile approach, the web browser provides the application-generic mechanism for user interaction and interaction with the user's host system, such as a PC, while the cooperating web browser server acts to provide the user access to the capabilities of the internet service. An internet service may adapt its presentation according to the connecting "User Agent", which may vary widely including from the typical PC web browser to an embedded browser in a cell phone to a UI-less "browser" within a dedicated digital imaging consumer electronics appliance.

A particular characteristic of the MPV Internet Profile is the ability to interact with assets local to the user's system without requiring the user to identify them by name or location. This is achieved by providing the location of local assets using a MPV manifest and associated data that is uploaded to the internet service.

1.4 Terms of Use

This section of the specification is descriptive and not intended to be complete nor definitive. Please refer to the definitive statement of licensing terms at the beginning of the MPV specification document for a precise and legal description.

The MPV specification is developed using an open process. The resulting specification is available from OSTA. No royalty is charged by OSTA for use of the specification. The overall desire is to develop a specification that is

not subject to separate licensing requirements or royalty. During the development process, the expectation is that all participants contribute their efforts and intellectual property without any expectation or requirement for compensation. However, OSTA does not warrant that the specification is not or will not be subject to such claims by other parties.

Chapter 2: MPV Internet Profile 1.0

2.1 MPV Internet Introduction

The MPV Internet Profile makes use of existing MPV specification ([MPVCore]) and combines them with additional specific requirements to define tightly the usage of these MPV profiles to guarantee interoperability between clients and servers that conform to the MPV Internet specification.

Conformance with the MPV Internet Profile specification is required for use of the MPV Logo on products.

The MPV Internet Profile introduces limited new schema and practices.

2.2 MPV Specification Practices

The MPV Internet Specification establishes three important practices.

1. All practices are qualified using the keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, if and where they appear in this document, are to be interpreted as described in [RFC2119].

   The keywords are classified into three imperative levels. All words at a given level have the same level of imperative.
   - Level 1: MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT
   - Level 2: SHOULD, SHOULD NOT
   - Level 3: RECOMMENDED, MAY, OPTIONAL

   For conformance testing, the keyword imperative levels are treated as warning levels, with the following meaning:
   - Level 1: Error -- MUST be fixed.
   - Level 2: Severe Warning -- SHOULD be fixed to enhance compatibility.
   - Level 3: Warning – RECOMMEND to be fixed. Not critical to compatibility.

2. All practices are classified as either Common, Writer, or Reader requirements. Common requirements are practices that are relevant to both Writers and Readers. Writer requirements identify specific MPV content that Writers must produce; Reader requirements identify specific behaviours that Readers must implement.

3. All practices are identified with a unique “requirement number” by which they may be referenced easily.

For example, a verification tools could output results and reference the requirement by number. Practices are identified using a prefix plus a number. This specification uses the letters "DO" plus the letter "C" for common, "W" for writer, or "R" for reader, plus the specification version number without the decimal separator as the prefix, as in PL100-83. A future revision of the specification might identify the same requirement by a different number if the number or order of the requirements were to change, e.g. DO110-85.

2.3 Formalities For Use of the MPV Internet Profile

The mechanism that MPV uses to add capabilities to the Core specification is the Profile. MPV Core sets out specific formalities to follow when a MPV Profile is used -- an MPV file must declare which profiles it implements and it must declare the namespaces of the profiles. This allows a processing application to quickly determine whether a given MPV file meets its expectations for processing.

PROFILE COMPONENTS

The MPV Internet Profile 1.0 makes use of the MPV Basic Profile 1.0 Specification **[MPVBasic]**.

The MPV Internet Profile 1.0 includes the schema and practices detailed by this document.

COMPATIBILITY

The MPV Internet Profile 1.0 is fully compatibile with the MPV framework established by **[MPVCore]**. Thus MPV files that implement the MPV Internet Profile 1.0 should be readable by most MPV-aware applications and devices that are not in conformance with the MPV Internet Profile.

Applications and devices that conform to the MPV Internet Profile 1.0 specification will provide the user a consistent user experience around the handling of MPVdocuments (manifests).

SCHEMA NAMESPACE

The MPV Internet Profile specification defines some schema, which uses the Document namespace.

Schema group	Namespace Identifier	Schema Location	Conventional Namespace Prefix
Document	http://ns.osta.org/mpv/internet/	lax/profiles/document/impl/internet.xsd	Mpvi:

The introductory schema information is expressed as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:mpvd="http://ns.osta.org/mpv/internet/1.0/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="qualified">
```

PROFILE IDENTIFIER

This information must be present in the Profile section of the MPV Manifest.

MPV Internet Profile Name	http://ns.osta.org/mpv/internet/1.0/

EXAMPLE

2.4 MPV Internet Profile Interaction Pattern

A typical interaction pattern for use of the MPV Internet Profile is as follows:

1. Service Discovery – the consumer starts using a web browser and selects a service they wish to use
    a. Discovery can be pre-determined, such as when a static web page delivered on a CD contains a link to a specific service provider
    b. Discovery can be dynamic, such as when the user navigates to find a service.
2. Conversation Request – the Web browser requests a conversation with the service provider
    a. Context-free request has no additional information than what is normally provided by a web browser request
    b. Context-aware request provides an asset manifest
3. Asset Manifest Upload
    a. The asset manifest may be uploaded as part of the conversation request
4. User Interaction
    a. Arbitrary user interaction may occur. Typically, this will include allowing the user to interact with the assets specified in the manifest, such as to select the assets that will be used by an online print fulfillment service and to collect customer order information.
    b. Internal to the service provider, this information is typically structured according to its own practices.
5. Asset Upload
    a. Any assets needed by the service are uploaded. This occurs by having the client initiate one or more asset upload operations.
    b. Asset upload may utilize a client-side upload application, if one is available on the consumer's device. The presence of this application can be detected by the service provider.
    c. Asset upload alternately may utilize any mechanism desired by the service provider. For example, on Microsoft Windows platforms, a small ActiveX control accessed within a web page may provide the upload functionality.
    d. If upload is interrupted, such as loss of an internet connection, out-of-band communication with the user (such as an email message) can lead them to return to the service provider and reinitiate asset upload. Successful completion of the operation may require that the asset leases have not expired.
6. Transaction Confirmation
    a. A confirmation of the transaction to the user is provided in a manner dependent on the service provider. Typically, a web page will be provided confirming the transaction, an email message may be sent as confirmation, or a receipt may be printed. This means for confirmation are not specified by the MPV Internet Profile.

2.5 MPV Internet Profile

DOW100-1 MPV Internet Profile. The MPV Internet profile MUST be implemented and declared.

This practice confirms that the manifest implemnts the MPV Internet profile.

Example:

```
<file:Manifest
        mpvd:writtenBy="http://www.mycompany.com/myapp/1.05/"
        mpvd:documentIdRef="ID000001"
        xmlns:file="http://ns.osta.org/manifest/1.0/"
        xmlns:mpv="http://ns.osta.org/mpv/1.0/"
        xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
        xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
        xmlns:mpvd="http://ns.osta.org/mpv/document/1.0/"
        xmlns:mpvi="http://ns.osta.org/mpv/internet/1.0/"
        xmlns:nmf="http://ns.osta.org/nmf/1.0/" >
  <nmf:Metadata>
    <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
      <ProfileBag>
          <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
          <Profile>http://ns.osta.org/mpv/presentation/1.0/</Profile>
          <Profile>http://ns.osta.org/mpv/music/1.0/</Profile>
          <Profile>http://ns.osta.org/mpv/internet/1.0/</Profile>
          <Profile>http://ns.osta.org/mpv/playlists/1.0/</Profile>
      </ProfileBag>
    </ManifestProperties>
  </nmf:Metadata>
...
</file:Manifest>
```

Chapter 3: Asset Manifest Content

MPV Manifest Usage

The MPV Internet Profile uses the "Basic Profile" of the MPV manifest specification [MPVBasic]. This provides for identifying assets and a selected subset of those assets.

To specify image-specific properties, such as width and height, MPV Internet Profile utilizes the BASIC_IMAGE_PARAM schema of the JPEG 2000 file format [J2Kp2]. These are included under the mpv:Metadata element of an asset.

Pathnames to files are specified using the <mpv:LastURL> element. Consistent with the MPV specification, in which MPV content is specified using the UTF-8 character encoding, these paths are specified using UTF-8. Furthermore, pathnames should also be either relative to the current location, such as "data/album.pvm", or use URL-encoded absolute paths, such as "file:///E:/data/album.pvm".

Sample Data

For purposes of this document, the following sample data will be used.

<table>
<tr><th>EXAMPLE</th><th>Example MPV Manifest</th></tr>
<tr><td>Assets</td><td>All these assets are stored on a CD.

/PICTURES/IMG_0001.JPG
/PICTURES/IMG_0002.JPG
/PICTURES/IMG_0003.JPG
/PICTURES/THUMB/IMG_0001.JPG
/PICTURES/THUMB/IMG_0002.JPG
/PICTURES/THUMB/IMG_0003.JPG
/PICTURES/SCREEN/IMG_0001.JPG
/PICTURES/SCREEN/IMG_0002.JPG
/PICTURES/SCREEN/IMG_0003.JPG

<table><tr><td>IMG_0001.JPG:</td><td>IMG_0002.JPG:</td><td>IMG_0003.JPG:</td></tr><tr><td>  </td><td>  </td><td>  </td></tr></table></td></tr>
<tr><td>MPV Manifest</td><td><?xml version="1.0" encoding="UTF-8"?>
<file:Manifest</td></tr>
</table>

```
        xmlns:file="http://ns.osta.org/manifest/1.0/"
        xmlns:mpv="http://ns.osta.org/mpv/1.0/"
        xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
        xmlns:nmf="http://ns.osta.org/nmf/1.0/" >
  <nmf:Metadata>
    <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
      <ProfileBag>
          <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
          <Profile>http://ns.osta.org/mpv/presentation/1.0/</Profile>
      </ProfileBag>
    </ManifestProperties>
  </nmf:Metadata>

  <mpvp:Album>
    <mpvp:Foreground>
       <mpv:StillRef mpv:idRef="ID000100"/>
       <mpv:StillRef mpv:idRef="ID000200"/>
       <mpv:StillRef mpv:idRef="ID000300"/>
    </mpvp:Foreground>
  </mpvp:Album>

  <!-- these are selected images from the set of known images -->
  <mpvb:MarkedAssets>
    <mpv:MarkList mpv:markType="selected">
       <mpv:StillRef mpv:idRef="ID000200"/>
       <mpv:StillRef mpv:idRef="ID000300"/>
    </mpv:MarkList>
  </mpvb:MarkedAssets>

  <mpv:AssetList>

    <!-- Still -->
    <mpv:Still mpv:id="ID000100">
      <mpv:ContentID>urn:osta-
org:mpv:dsig:md5:all:3F886AEFA3B340da971BAF09B17DBC122</mpv:ContentID>
      <mpv:LastURL>PICTURES/IMG_0001.JPG</mpv:LastURL>
      <mpv:Metadata>
         <BASIC_IMAGE_PARAM xmlns="http://www.jpeg.org/jpx">
            <BASIC_IMAGE_INFO>
               <IMAGE_SIZE>
                  <WIDTH>1600</WIDTH>
                  <HEIGHT>1200</HEIGHT>
               </IMAGE_SIZE>
            </BASIC_IMAGE_INFO>
         </BASIC_IMAGE_PARAM>
      </mpv:Metadata>
      <mpv:Rendition mpv:renditionUsage="thumbnail">
         <mpv:StillRef mpv:idRef="ID000101"/>
      </mpv:Rendition>
       <mpv:Rendition mpv:renditionUsage="screen">
         <mpv:StillRef mpv:idRef="ID000102"/>
      </mpv:Rendition>
    </mpv:Still>

    <mpv:Still mpv:id="ID000101">
      <mpv:ContentID>urn:osta-
org:mpv:dsig:md5:all:89886AEFA3B340da971BAF09B17DBC342</mpv:ContentID>
      <mpv:LastURL>PICTURES/THUMB/IMG_0001.JPG</mpv:LastURL>
    </mpv:Still>

    <mpv:Still mpv:id="ID000102">
      <mpv:ContentID>urn:osta-
org:mpv:dsig:md5:all:45886AEFA3B340da971BAF09B17DBC567</mpv:ContentID>
      <mpv:LastURL>PICTURES/SCREEN/IMG_0001.JPG</mpv:LastURL>
    </mpv:Still>

    <!-- Still -->
    <mpv:Still mpv:id="ID000200">
      <mpv:ContentID>urn:osta-
org:mpv:dsig:md5:all:F9886AEFA3B340da971BAF09B17DBC199</mpv:ContentID>
      <mpv:LastURL>PICTURES/IMG_0002.JPG</mpv:LastURL>
```

	<mpv:Metadata> <BASIC_IMAGE_PARAM xmlns="http://www.jpeg.org/jpx"> <BASIC_IMAGE_INFO> <IMAGE_SIZE> <WIDTH>1600</WIDTH> <HEIGHT>1200</HEIGHT> </IMAGE_SIZE> </BASIC_IMAGE_INFO> </BASIC_IMAGE_PARAM> </mpv:Metadata> <mpv:Rendition mpv:renditionUsage="thumbnail"> <mpv:StillRef mpv:idRef="ID000201"/> </mpv:Rendition> <mpv:Rendition mpv:renditionUsage="screen"> <mpv:StillRef mpv:idRef="ID000202"/> </mpv:Rendition> </mpv:Still> <mpv:Still mpv:id="ID000201"> <mpv:ContentID>urn:osta- org:mpv:dsig:md5:all:C7886AEFA3B340da971BAF09B17DBC326</mpv:ContentID> <mpv:LastURL>PICTURES/THUMB/IMG_0002.JPG</mpv:LastURL> </mpv:Still> <mpv:Still mpv:id="ID000202"> <mpv:ContentID>urn:osta- org:mpv:dsig:md5:all:A8886AEFA3B340da971BAF09B17DBC502</mpv:ContentID> <mpv:LastURL>PICTURES/SCREEN/IMG_0002.JPG</mpv:LastURL> </mpv:Still> <!-- Still --> <mpv:Still mpv:id="ID000300"> <mpv:ContentID>urn:osta- org:mpv:dsig:md5:all:23886AEFA3B340da971BAF09B17DBC161</mpv:ContentID> <mpv:LastURL>PICTURES/IMG_0003.JPG</mpv:LastURL> <mpv:Metadata> <BASIC_IMAGE_PARAM xmlns="http://www.jpeg.org/jpx"> <BASIC_IMAGE_INFO> <IMAGE_SIZE> <WIDTH>1600</WIDTH> <HEIGHT>1200</HEIGHT> </IMAGE_SIZE> </BASIC_IMAGE_INFO> </BASIC_IMAGE_PARAM> </mpv:Metadata> <mpv:Rendition mpv:renditionUsage="thumbnail"> <mpv:StillRef mpv:idRef="ID000301"/> </mpv:Rendition> <mpv:Rendition mpv:renditionUsage="screen"> <mpv:StillRef mpv:idRef="ID000302"/> </mpv:Rendition> </mpv:Still> <mpv:Still mpv:id="ID000301"> <mpv:ContentID>urn:osta- org:mpv:dsig:md5:all:93886AEFA3B340da971BAF09B17DBC390</mpv:ContentID> <mpv:LastURL>PICTURES/THUMB/IMG_0003.JPG</mpv:LastURL> </mpv:Still> <mpv:Still mpv:id="ID000302"> <mpv:ContentID>urn:osta- org:mpv:dsig:md5:all:35886AEFA3B340da971BAF09B17DBC584</mpv:ContentID> <mpv:LastURL>PICTURES/SCREEN/IMG_0003.JPG</mpv:LastURL> </mpv:Still> </mpv:AssetList> </file:Manifest>
MPV Manifest	This will vary on each system and data source. For this example, it is:

Location	On the CD: /ALBUM.PVM CD accessed via the computer: E:/ALBUM.PVM (E drive is currently providing access to the disc.)

Types of MPV Internet Profile Services

There is one primary kind of MPV Internet Profile services with plain HTTP interaction (context-free)

There are many useful and possible means of discovery of a MPV Internet Profile provider. For example, the following methods are all practical and feasible.

1. The provider is at a fixed URL, which is authored into a static webpage on a CD.
2. A fixed URL is authored into a static webpage on a CD, but when used, it redirects the browser to a provider.
3. A browser-based playback applet for asset manifests has a fixed URL that it uses for provider-based access
4. A locator service can query a directory for a range of services that offer a given functionality and are also providers. It presents the user a webpage with links to the desired service.
5. The user directly types the target service provider URL, such as http://www.myphotoservicessite.com/mpv/, using the knowledge that most MPV-capable websites support this entry point URL for internet services.

A compliant service provider will identify at least the following information about bindings through which to access the service.

(1) Either or both of:
   a. HTTP (Context-Free) Conversation Request
   b. MPV Context-Aware Conversation Request

This information is provided in the following manner:

SPEC	Service Providers
Service Information	Not specified.
Service Identification	Not specified.
MPV Internet Profile plain HTTP (Context-Free) service	• the *accessPoint* provides the entry point for the initial MPV Internet Profile conversation
MPV Internet Profile MPV Context-Aware Conversation	• the *accessPoint* provides the entry point for the initial MPV Internet Profile conversation

Service Protocol Implementation

A internet service that implements the MPV Internet Profile interface has relatively few protocol items to conform to.

Security and Privacy

Any implementation of the MPV Internet Profile specification includes an exchange of information between the user, user's system (client) and one or more servers. The security and privacy of this information is important to consider and handle appropriately.

Much of the information that is provided during interaction is not addressed or controlled in any way by the Internet Profile specification – this is the information that is exchanged between initial upload of the manifest and the upload request for specific assets. This information might include order information about specific products and user identifiable information, such as name, address, credit card information, etc. It is not in the scope of the MPV Internet Profile specification to specify how this information is to be collected or handled; although naturally it is recommended to take careful steps to protect a user's privacy and the security of this information.

Manifest Upload and Download

Relevant to the MPV Internet Profile specification is the privacy and security of a user's MPV manifest. The manifest contains information about assets on the user's system, and servers with this information can initiate uploads of the user's assets, possibly without their awareness. This means that the manifest information should always be treated in a secure manner when transmitted over the internet. Therefore, a MPV Internet Profile client and server MUST always communicate over https **[SSL]** connections whenever manifest information is present in the communication and not otherwise protected (e.g. encrypted).

Thus, as specified in Chapter 0, MPV Internet Profile Conversation Request, the MPV Internet Profile manifest upload MUST occur over a https connection, and as specified in Chapter 0, Asset Transfer, the upload request manifest MUST also be provided over https. Note that for purposes of this specification, all interaction is specified as HTTP protocol. https implements HTTP over SSL.

Manifest Information Cached on Webpages

A more subtle situation is the common practice of caching information on webpages during a user's interaction with a website by passing it along as hidden (or even visible) variables on webpages. This practice is convenient, since it avoids storing temporary information on the webserver and helps enhance the stateless interaction with a webserver. It will likely be a common practice when implementing MPV Internet Profile servers because of the need to maintain manifest information between the initial manifest upload, the use of the manifest information durin user interaction to construct the order, and generation of the upload request manifest (see Chapter 0, Asset Transfer).

While this aspect of a MPV Internet Profile server implementation is beyond the scope of the MPV Internet Profile specification itself because it involves implementation practices specific only to each server, the MPV Internet Profile implementor is urged to keep take steps to keep MPV manifest information secure and private.

Passing along MPV manifest information from webpage to webpage can be done securely in several ways. For example, pages that contain MPV manifest information as hidden variables could use a https connection. Alternately, the server could use a server-side secret to encrypt the manifest information before storing the information in hidden variables. When needed, the manifest information is decrypted by the server and used to generate the upload request manifest. Other techniques may be possible.

MPV Internet Profile Conversation Request

A MPV Internet Profile conversation request is what occurs when the web browser begins a conversation with the MPV Internet Profile provider. The conversation is initiated by an HTTP GET or HTTP POST request to the MPV Internet Profile provider by the web browser.

Conversation Request URL Path Convention

A MPV Internet Profile provider can accept conversation requests at any URL. However, this specification provides a distinguished path:

SPEC	Conversation Request URL Path Convention
Introduction	A conventional URL that MAY be used by MPV Internet Profile provider implementations. Consistent URL design allows synthesis of a possible MPV Internet Profile Conversation Request URL given only a domain name, either by users or programmatically.
URL convention	**`{conventional domain root syntax}/mpv/`**
Parameters	• **`{conventional domain root syntax}`** is the syntax that is conventional for a domain root in a given situation. • **`mpv`** is the conventional distinguished path for conversation requests
Discussion	Note that the path completes with the "/" character. This allows simple implementation in many different web server environments by utilizing the server ability to activate an unnamed default handler which may vary in name according to the implementation technology used.

EXAMPLE	Conversation Request URL Path Convention
URL	www.somephotoprints.com/mpv/
URL	www.irgendeinefotoaufnahme.co.de/mpv/
URL	www.somefotoprints.co.jp/mpv/
URL	www.somephotoprints.com/mpv/?MPV_MANIFEST_POST_PAGE=https%3A%2F%2FDATA%2FMPV OST_9234852883406.HTM

Plain HTTP (Context-Free) Conversation Request

At its simplest, no MPV-related context information is provided as part of the HTTP GET/POST request to the MPV Internet Profile provider. Not considered here is typical context provided by web browsers as part of the basic HTTP protocol [HTTP1.1], such as the URL of the request, user agent, accepted content types, and cookie header information.

SPEC	Context-Free Conversation Request
Introduction	A context-free conversation request MUST conform to HTTP GET/POST usage for common web browsers and not contain any context.
HTTP Protocol (Requestor to MPV Internet Profile Provider)	**HTTP 1.* GET or POST request** No MPV Internet Profile-specific GET or POST content.

The follow example illustrates a typical context-free conversation request.

EXAMPLE	Typical Context-Free Conversation Request
HTTP Protocol (Requestor to MPV Internet Profile Provider)	**GET /mpv/ HTTP/1.1** **Accept: text/*, image/gif, image/jpg, */*** **Accept-Encoding: gzip** **Accept-Language: en, en_US** **Connection: Keep-Alive** **Cookie: userID=id456578** **Host: www.somephotoservicessite.com** **Referer: http://www.somephotoservicessite.com/index.html** **User-Agent: Mozilla/4.7 [en] (Win98; U)**
Explanation	• The GET line is the request line that comprises three fields: 1. a method: The GET method indicates that the server is supposed to return an entity. 2. a request-URI (Universal Resource Identifier). The "/mpv/" indicates the path mpv under the root of the document system on the server and the default handler for that path (the trailing "/" indicates the default handler for the "/mpv" path), and 3. HTTP protocol version: 1.1 in this case. • The Connection line is the optional Connection header informs the server that the browser would like to leave the connection open after the response. • The User-Agent line is the optional User-Agent header that identifies the kind of browser that is sending the request, its version, and its operating system. • The Accept headers specify the type, language, and encoding for the returned entity that the browser would prefer to receive from the server.
HTML Example	<!—Context-Free Client-to-MPV Internet Profile provider Conversation Request --> <HTML> <Head> </Head> <Body> Context-Free Conversation Request </Body> </HTML>

This type of conversation does not need to be specified further other than to require that the MPV Internet Profile provider conform to established web standards for interaction with a web browser. No context information is available about the consumer's assets or interests and all such information must be provided explicitly by the consumer using typical web browser-based interactive interaction patterns. This situation is not discussed further.

The MPV Internet Profile specification permits context information to be provided after a context-free conversation has begun. Mechanisms to do so are discussed elsewhere. Once context has been provided, a context-aware conversation can proceed, as discussed below.

Context-Aware Conversation Request

Much more interesting is when the conversation request includes information that provides a MPV-related context for the conversation. This context is centered on an asset manifest which describes the assets to be referenced during the interaction. In MPV Internet Profile 1.0, the only manifest type supported is a MPV manifest **[MPVBasic]**. Other types of manifests may be supported in future MPV Internet Profile revisions.

A Context-Aware conversation is able to provide a powerful user experience in which the MPV Internet Profile provider interacts with the user using the assets identified by the context-aware conversation request, but without first uploading them. For example, the MPV Internet Profile provider can present the user with web pages showing thumbnail images of all the assets described by the manifest, allowing the user to select images to use by seeing the images embedded within the webpage. Remarkably, these images can be referenced in their actual location, typically on the user's local file system. In this fashion, the user interaction is guided by the MPV Internet Profile provider while the assets themselves remain local to the user. Only the selected files, in the selected resolutions, are then uploaded or further processed in accordance to the user's choices. Using standard MPV capabilities, it is even possible to determine robustly that a given asset has already been uploaded and need not be uploaded again.

In the conversation request, access to the asset manifest may be provided in two ways:

Case 1: Integrated Manifest Upload, consisting of the asset manifest and related information

Case 2: Separate Manifest Upload, consisting of the location of a Manifest Upload Page, a webpage that will transfer the asset manifest and related information to the MPV Internet Profile provider when it is loaded. This allows a server to request the information on demand.

Integrated Manifest Upload

One form of context-aware conversation request integrates upload of the asset manifest and related information as part of the conversation request by using the mechanisms specified in detail in Chapter 0, MPV Internet Profile Manifest Upload. This approach will typically provide the best user interaction performance because no further steps are required in order to access the asset manifest. This form of manifest upload is fully specified in Chapter 0.

Separate Manifest Upload

Another form of context-aware conversation request provides the MPV Internet Profile provider with the location of a webpage that implements integrated manifest upload. The Manifest Upload Page is a web page that arranges to upload the asset manifest and related information to a specified URL when the page is loaded. Typically, this webpage location will be a URL to a temporary file on a webserver to which the manifest was previously uploaded. It will NOT be a path in the PC or device's local filesystem because some recent browsers have decided that a redirect from a server to a local page is a security risk.

This approach allows the first server which receives the conversation request to redirect the browser using a standard HTTP 302 reply or any other type of redirect mechanism. The target server receives the redirected page along with the pickup location for the MPV manifest; it can then fetch the MPV manifest from the first server at the location specified.

This approach to redirects is to be favored over approaches (which are viable, just slower in performance) in which redirects are implemented by downloading a HTML document that includes the MPV manifest and then rePOSTs itself automatically to a new server. Repeatedly downloading and uploading the MPV manifest to implement redirects decreases the performance of a redirect on slow connections. It also requires on-going use of https (SSL) connections to protect the privacy of the manifest information.

The specification is as follows.

SPEC	Conversation Request with Separate Manifest Upload
Introduction	• The Manifest Upload Page MUST be specified via a HTTP GET request. • Other arguments passed on the conversation request's initial GET request MUST be passed along to the manifest upload page; this allows for communication between the two client pages. • The Manifest Upload Page must be reachable within typically allowed security limitations of web browsers. o Some web browsers require that the upload page is NOT local to the web browser client, as a redirect from a web server to a local page is considered a security risk.
HTTP Protocol (Requestor to MPV Internet Profile Provider) - GET protocol for requesting a conversation	**GET /{request-URI}?MPV_MANIFEST_POST_PAGE={page-URI}{other content}** **HTTP/1.{0|1}** **{other HTTP content}**
Parameters	• **{request-URI}** is able to accept Conversation Requests with Separate Manifest Upload • **MPV_MANIFEST_POST_PAGE** is the name of the field • **{page-URI}** is the value of **MPV_MANIFEST_POST_PAGE** field • **{other content}** is additional other content in the URI. This other content is passed along to the **{page-URI)** page on invocation. Other content uses the '&' character to separate fields/value pairs per the URI specification.

The conversation request specifies a MPV Manifest Post Page. This webpage, when loaded, which post the MPV manifest to a specified location. The page invocation is as follows:

SPEC	Manifest Upload Page Invocation
Introduction	A Manifest Upload Page conforms to the specifications for manifest upload as specified in Chapter 0, MPV Internet Profile Manifest Upload. However, in addition: • the manifest upload page MUST accept an argument that specifies the URL to which the manifest should be uploaded. • the manifest upload page SHOULD automatically submit the manifest to that location when the page is loaded. This provides the smoothest user experience.

Manifest Post Page invocation spec	The manifest upload page MUST be invoked with a target URL to which to POST the manifest. The page is invoked using the following syntax:
	{MPV_MANIFEST_POST_PAGE}?MPV_MANIFEST_POST_URL={URI}{other content}
	
	Using HTTP protocol returned from server to client, the invocation looks like this:
	
	`HTTP/1.1 302 FOUND`
	`...`
	`Location:` **{MPV_MANIFEST_POST_PAGE}?MPV_MANIFEST_POST_URL={URI}{other content}**
	`...`

Parameters	• **{MPV_MANIFEST_POST_PAGE}** is the value specified in the conversation request.

	• **MPV_MANIFEST_POST_URL** is a field name • **{URI}** is any valid URI that will accept the manifest upload conforming to the MPV Internet Profile specification. • **{other content}** is additional other content in the URI specified in a syntactically correct manner. Other content must conform to URI syntax for field separation using the '&' character.
Implementation and Behaviour Notes	Practically speaking, the manifest post page must use JavaScript or similar client-side processing capability in order to process the invocation arguments and to initiate the form submit action. The page SHOULD automatically submit the form to the target URI when the page is loaded; this typically provides the best user experience.

EXAMPLE

The following examples implement the specification in the context of using a Microsoft Internet Explorer 4.0 or newer web browser on Microsoft Windows-based OS.

EXAMPLE	Conversation Request with Manifest Upload Page

HTTP Protocol (Requestor to MPV Internet Profile Provider)
 	
- GET protocol
	`GET /mpv/?MPV_MANIFEST_POST_PAGE=https%3A%2F%2FDATA%2FMPV_POST_9234852883406.HTM HTTP/1.1`
	`Accept: text/*, image/gif, image/jpg, */*`
	`Accept-Encoding: gzip`
	`Accept-Language: en, en_US`
	`Connection: Keep-Alive`
	`Cookie: userID=id456578`
	`Host: www.somephotoservicessite.com`
	`Referer: www.somelocatorservice.com/locator.asp`
	`User-Agent: Mozilla/4.7 [en] (Win98; U)`

Explanation	• The **MPV_MANIFEST_POST_PAGE** is a field whose value is **https://DATA/MPV_POST_9234852883406.HTM** (shown url-encoded according to the HTTP Proto • The referrer is a service locator who cached the initial MPV manifest upload from the client and now is providing it to the service to which it is directing the user.

HTML Example
 	
Initial page
	`<!–Context-aware Client-to-MPV Internet Profile provider Conversation Request -->`
	`<HTML>`
	`<Head>`
	`</Head>`
	`<Body>`
	`<a href="https://www.somephotoservicessite.com/mpv/?MPV_MANIFEST_POST_PAGE=https%3A%2F%2FDA 2FMPV_POST_9234852883406.HTM">`
	`Context-Aware Conversation Request</a>`
	`</Body>`
	`</HTML>`

The manifest upload page conforms to the specifications for manifest upload as specified in Chapter 0, MPV Internet Profile Manifest Upload, but also arranges to upload the manifest when the page is loaded and to a specified destination.

EXAMPLE	Manifest Upload Page

Typical invocation	The client is redirected to the upload page by the MPV Internet Profile provider:
	
	`HTTP/1.1 302 Moved Temporarily`
	`Date: Mon, 06 Jul 2002 20:54:26 GMT`
	`Server: Apache/1.3.6 (Unix)`
	`Last-Modified: Mon, 06 Jul 2002 20:52:11 GMT`
	`ETag: "2f5cd-964-381e1bd6"`

	`Location: https://DATA/MPV_POST_9234852883406.HTM?\`
	`MPV_MANIFEST_POST_URL=www.somephotoservicessite.com/mpv/manifest/`
	`Accept-Ranges: bytes`
	`Content-length: 327`
	`Connection: close`
	`Content-type: text/html`

Assumptions	In this example, the MPV_POST_9234852883406.HTM page is generated on-the-fly by the service locator server when it receives the initial MPV-context-aware conversation request to include the MPV manifest and other information as a string values that can be reposted to a new destination.
HTML Example MPV_POST.HTM	

```
<HTML>
  <HEAD>
   <SCRIPT language="JavaScript">

    // the MPV Manifest encoded as a JavaScript string
    function get_Manifest()
    {
     var xml;
       // quote- and backslash-escaped string
     xml="<?xml version=\"1.0\" encoding=\"UTF-8\"?>
          <file:Manifest
          xmlns:file=\"http://ns.osta.org/manifest/1.0/\"
          xmlns:file=\"http://ns.osta.org/mpv/1.0/\"
          xmlns:file=\"http://ns.osta.org/presentation/1.0/\"
          xmlns:file=\"http://ns.osta.org/nmf/1.0/\">
     ... lots more here ...
          </file:Manifest>";
     return xml;
    }

    function get_ManifestLoc()
    {
     return "file:///E:/ALBUM.PVM";
    }

    function get_ManifestRoot()
    {
     return "file:///E:/";
    }

    // extract form action URL from the invocation path arguments (“search
string”)
    function get_SubmitDest()
    {
       var dest;
     dest = location.search;
     dest = dest.substring(1,dest.length);      //remove the question mark
       // extract the value of MPV_MANIFEST_POST_URL
     dest = dest.replace("/.*MPV_MANIFEST_POST_URL=\([^&]*\)(&.*)?$/\1/",dest);
     dest = unescape(dest);
     return dest;
    }

    function load_actions()
    {
       // we know that the album file is in the same directory as the html
page
     form1.MPV_MANIFEST_LOCATION.value=get_ManifestLoc();
     form1.MPV_MANIFEST_ROOT_PATH.value=get_ManifestRoot();
     form1.MPV_MANIFEST_CONTENT.value=get_Manifest();
     form1.action=get_SubmitDest();
```

```
     form1.submit()
    }
   </SCRIPT>

  </HEAD>
  <BODY bgcolor="#FFFFFF"  onLoad="load_actions()" >
     <FORM ID="form1" NAME="form1"
           ENCTYPE="multipart/form-data"
           METHOD=POST>
          <INPUT TYPE=SUBMIT VALUE="Order Prints Online">
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_LOCATION
NAME=MPV_MANIFEST_LOCATION>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_CONTENT
NAME=MPV_MANIFEST_CONTENT>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_ROOT_PATH
NAME=MPV_MANIFEST_ROOT_PATH>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_MANIFEST_MIME_TYPE
NAME=MPV_UPLOAD_MANIFEST_MIME_TYPE VALUE=”application/vnd.osta-
org.upload.mpv+xml”>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_RQST_FILE_EXT
NAME=MPV_UPLOAD_RQST_FILE_EXT VALUE=”mpvumpv”>
          <INPUT TYPE=HIDDEN
ID=MPV_INTERNET_PROFILE_VERNAME=MPV_INTERNET_PROFILE_VERVALUE="1.0">
     </FORM>
  </BODY>
</HTML>
```

For reference, we also provide an example of a MPV manifest post page that would be implemented local to the browser client. It constructs the MPV manifest information from locally available information. This approach may be useful for creating the initial conversation request to a webserver, but recent releases of some web browsers have disallowed a website from redirecting back to a client-local page due to security risks.

HTML Example

MPV_POST.HTM

```
<HTML>
  <HEAD>
   <SCRIPT language="JavaScript">

    // the MPV Manifest encoded as a JavaScript string
    function get_Manifest()
    {
     var xml;
     xml="<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
     xml+="<file:Manifest\n";
     xml+="xmlns:file=\"http://ns.osta.org/manifest/1.0/\"\n";
     xml+="xmlns:file=\"http://ns.osta.org/mpv/1.0/\"\n";
     xml+="xmlns:file=\"http://ns.osta.org/presentation/1.0/\"\n";
     xml+="xmlns:file=\"http://ns.osta.org/nmf/1.0/\">\n";
     ... lots more here ...
     xml+="</file:Manifest>\n";
     return xml;
    }

    function get_ManifestLoc()
    {
       var loc;
     loc = get_ManifestRoot();
     loc += "ALBUM.PVM";
     return loc;
    }

    function get_ManifestRoot()
    {
       var loc;
     loc = window.location;
     loc = loc.replace ("/\(.*\)\/.*(\?.*)?/\1\//"); //remove fname and all after
```

```
q-mark
     return loc;
    }

    // extract form action URL from the invocation path arguments (“search
string”)
    function get_SubmitDest()
    {
       var dest;
     dest = location.search;
     dest = dest.substring(1,dest.length);      //remove the question mark
       // extract the value of MPV_MANIFEST_POST_URL
     dest = dest.replace("/.*MPV_MANIFEST_POST_URL=\([^&]*\)(&.*)?$/\1/",dest);
     dest = unescape(dest);
     return dest;
    }

    function load_actions()
    {
       // we know that the album file is in the same directory as the html
page
     form1.MPV_MANIFEST_LOCATION.value=get_ManifestLoc();
     form1.MPV_MANIFEST_ROOT_PATH.value=get_ManifestRoot();
     form1.MPV_MANIFEST_CONTENT.value=get_Manifest();
     form1.action=get_SubmitDest();
     form1.submit()
    }
   </SCRIPT>

  </HEAD>
  <BODY bgcolor="#FFFFFF"  onLoad="load_actions()" >
     <FORM ID="form1" NAME="form1"
           ENCTYPE="multipart/form-data"
           METHOD=POST>
          <INPUT TYPE=SUBMIT VALUE="Order Prints Online">
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_LOCATION
NAME=MPV_MANIFEST_LOCATION>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_CONTENT
NAME=MPV_MANIFEST_CONTENT>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_ROOT_PATH
NAME=MPV_MANIFEST_ROOT_PATH>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_MANIFEST_MIME_TYPE
NAME=MPV_UPLOAD_MANIFEST_MIME_TYPE VALUE=”application/vnd.osta-
org.upload.mpv+xml”>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_RQST_FILE_EXT
NAME=MPV_UPLOAD_RQST_FILE_EXT VALUE=”mpvumpv”>
          <INPUT TYPE=HIDDEN
ID=MPV_INTERNET_PROFILE_VERNAME=MPV_INTERNET_PROFILE_VERVALUE="1.0">
     </FORM>
  </BODY>
</HTML>
```

Typically, to access this page, the web server must issue a redirect command to the page using typical web browser redirection commands. Recall that for security reasons, this page must be accessed via https (SSL).

MPV Internet Profile Manifest Upload

The interaction with the user can be much richer and easier-to-use when the MPV Internet Profile provider has access to an asset manifest. This manifest can be uploaded according to the specifications of this chapter. In MPV Internet Profile 1.0, the only manifest type supported in a MPV manifest **[MPVBasic]**.

A MPV Internet Profile provider with access to a manifest is able to carry on a context-aware conversation with the user, enabling a powerful user experience in which the MPV Internet Profile provider uses the assets identified by the context-aware conversation request in the user interaction, but without first uploading the assets. For example, the MPV Internet Profile provider can present the user with web pages showing thumbnail images of all the assets described by the manifest, allowing the user to select images to use by seeing the images embedded within the webpage. Remarkably, these images can be referenced in their actual location, typically on the user's local file system. In this fashion, the user interaction is guided by the MPV Internet Profile provider while the assets themselves remain local to the user. Only the selected files, in the selected resolutions, are then uploaded or further processed in accordance to the user's choices. Using standard MPV capabilities, it is even possible to determine robustly that a given asset has already been uploaded and need not be uploaded again.

In the manifest upload, the asset manifest is uploaded using the HTTP protocol for form data upload **[HTTP1.1][FileUpload]**. A primary use case is for manifest upload is using a webpage that is pre-generated and contains the manifest encoded in its body or loaded from a file of a known name using web browsers capable of processing XML content, such as recent Microsoft Internet Explorers. This is a common situation for digital image storage media, such as recordable CDs, which contain a pre-generated webpage that the user may interact with to access the contents of the disc. It may also occur when using web-browser based applications.

Manifest Security

It is very important to maintain the privacy of a user's manifest. The manifest contains information about assets on the user's system, and servers with this information can initiate uploads of the user's assets, possibly without their awareness. This means that the manifest information should always be treated in a secure manner when transmitted over the internet. Therefore, a MPV Internet Profile client and server MUST always communicate over https **[SSL]** connections whenever manifest information is present in the communication.

Thus the MPV Internet Profile manifest upload MUST occur over a https connection and as specified in Chapter 0, Asset Transfer, the upload request manifest MUST also be provided over https. Note that for purposes of this specification, all interaction is specified as HTTP protocol. https implements HTTP over SSL.

Manifest Content

The uploaded manifest MUST contain an <mpv:AssetList> element and subelements. This defines the set of assets to be utilized in the conversation. In addition, the uploaded manifest MAY contain an <mpvb:MarkedAssets> element and <mpv:MarkList mpv:markType="selected"> subelement. This allows the client to specify a selected set of assets to be pre-selected as part of subsequent interactions with the user. Note that assets may also be selected by using one or more occurances of the MPV_SELECTED form variable.

When no assets are preselected through either <mpvb:MarkedAssets> or the MPV_SELECTED field, this is interpreted to be that all assets are preselected. It is not possible to specify that no assets are preselected.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
        xmlns:file="http://ns.osta.org/manifest/1.0/"
        xmlns:mpv="http://ns.osta.org/mpv/1.0/"
        xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
        xmlns:nmf="http://ns.osta.org/nmf/1.0/" >
...

  <!-- these are selected images from the set of known images -->
  <mpvb:MarkedAssets>
    <mpv:MarkList mpv:markType="selected">
       <mpv:StillRef mpv:idRef="ID000200"/>
       <mpv:StillRef mpv:idRef="ID000300"/>
    </mpv:MarkList>
  </mpvb:MarkedAssets>

  <mpv:AssetList>
...
  </mpv:AssetList>

</file:Manifest>
```

Manifest Upload Encoded as multipart/form-data

The MPV Internet Profile Manifest Upload protocol using the form data protocol based on HTTP POST and the "multipart/form-data" encoding MUST be supported.

SPEC	multipart/form-data Manifest Upload	
Introduction	• Manifest Upload MUST utilize a HTTP POST request. • Data is encoded according to multipart/form-data **[HTTP1.1][FileUpload]** • The following information MUST be provided: o MPV Manifest o Root Path to use for relative pathnames found in the manifest • The following information SHOULD be provided: o File-system location of the Manifest o MPV Internet Profile specification version of the upload protocol • The following information MAY be provided: o Mime type of the Asset Upload Request Manifest document. The default value is specified in Section 0, Asset Upload Client Identification. o File extension of the Asset Upload Request Manifest document. The default value is specified in Section 0, Asset Upload Client Identification.	

HTTP Protocol (Requestor to MPV Internet Profile Provider)		`POST /{request-URI} HTTP/1.1`
		`{other HTTP content}`
		`Content-type: multipart/form-data, boundary={boundary identifier}`
		
		`--{boundary identifier}`
		`content-disposition: form-data; name="MPV_MANIFEST_LOCATION"`
		
		`{manifest-file-path}`
		`--{boundary identifier}`

	content-disposition: form-data; name="**MPV_MANIFEST_ROOT_PATH**" **{root-path}** --{boundary identifier} content-disposition: form-data; name="**MPV_UPLOAD_MANIFEST_MIME_TYPE**" **{upload-manifest-mime-type}** --{boundary identifier} content-disposition: form-data; name="**MPV_UPLOAD_RQST_FILE_EXT**" **{upload-rqst-file-ext}** --**{boundary identifier}** content-disposition: form-data; name="**MPV_SELECTED**" **{selected-asset-id}** --**{boundary identifier}** content-disposition: form-data; name="**MPVInternet Profile_VER**" **{ver}** --**{boundary identifier}** content-disposition: form-data; name="**MPV_MANIFEST_CONTENT**" **{manifest-content}** --**{boundary identifier}**--
Parameters	• **{request-URI}** is able to accept Conversation Requests with Manifest Upload • **multipart/form-data** is the content type in accordance to HTTP. • **{boundary-identifier}** is generated by the client in accordance to HTTP. • **MPV_MANIFEST_LOCATION** is the name of the field specifying a root-relative fully qualified path to the manifest file location or other suitable identifier. The MPV_MANIFEST_LOCATION value is passed back to the requestor by the MPV Internet Profile provider as the <mpv:LastURL> element of the <mpv:ManifestLink> in the Asset Upload Request. If no manifest location provided, asset upload is not possible because the upload client cannot access the manifest that contain the MPV assets referenced in the upload request. • **{manifest-file-path}** is the value of **MPV_MANIFEST_LOCATION** field. In most cases, this will be a root-relative fully qualified path to the manifest file location. Exceptionally, it may be an arbitrary identifier that allows the client to identify the manifest; this might be appropriate for manifests stored in memory and never written to file, for example. It is a URL-encoded string. • **MPV_MANIFEST_ROOT_PATH** is the root path that is used with relative pathnames in the manifest to access client-side assets. • **{root-path}** is the value of **MPV_MANIFEST_ROOT_PATH** field. The path terminates in a slash "/". It may use any transport (e.g. file, http). It is a URL-encoded string. • **MPV_UPLOAD_MANIFEST_MIME_TYPE** is the field used to specify the mime type of the upload request manifest document. • **{upload-manifest-mime-type}** is the value **of MPV_UPLOAD_MANIFEST_MIME_TYPE** field. This string is used as the type of the upload request manifest document. • **MPV_UPLOAD_RQST_FILE_EXT** is the field used to specify the file extension of the upload request manifest document. • **{upload-rqst-file-ext}** is the value **of MPV_UPLOAD_RQST_FILE_EXT** field. This string is used as the file extension of the upload request manifest document. • **MPV_SELECTED** is the field used to specify an ID of a MPV asset that is pre-selected. More than one occurance of MPV_SELECTED fields may be present, allowing multiple assets to be pre-selected. If no assets are preselected, they are all preselected. Specifying a selected asset via a form variable is an alternative to specifying a selected asset via the

	<mpvb:MarkedAssets> element. • **{selected-asset-id}** is the value of the **MPV_SELECTED** field. This string must correspond to the asset id of an asset in the manifest content document. • **MPV_MANIFEST_CONTENT** is the name of the field specifying the MPV manifest content. • **{manifest-content}** is the value of **MPV_MANIFEST_CONTENT** field. • **MPV_INTERNET_PROFILE_VER**is the name of the field specifying the MPV Internet Profile version number implemented. • **{ver}** is "1.0"

EXAMPLE

The following example illustrates uploading the MPV manifest and related information to the MPV Internet Profile provider.

EXAMPLE	**multipart/form-data Manifest Upload**	
HTTP Protocol (Requestor to MPV Internet Profile Provider)		POST /mpv/ HTTP/1.1 Accept: text/*, image/gif, image/jpg, */* Accept-Encoding: gzip Accept-Language: en, en_US Connection: Keep-Alive Cookie: userID=id456578 Host: www.somephotoservicesco.com Referer: http://www.somephotoservicesco.com/index.html User-Agent: Mozilla/4.7 [en] (Win98; U) Content-type: multipart/form-data, boundary=AaB03x --AaB03x content-disposition: form-data; name="MPV_MANIFEST_LOCATION" file:///E:/ALBUM.PVM --AaB03x content-disposition: form-data; name="MPV_MANIFEST_ROOT_PATH" file:///E:/ --AaB03x content-disposition: form-data; name="MPV_UPLOAD_MANIFEST_MIME_TYPE" application/vnd.osta-org.upload.mpv+xml --AaB03x content-disposition: form-data; name="MPV_UPLOAD_RQST_FILE_EXT" mpvumpv --AaB03x content-disposition: form-data; name=" MPV_INTERNET_PROFILE_VER" 1.0 --AaB03x content-disposition: form-data; name="MPV_MANIFEST_CONTENT" <?xml version="1.0" encoding="UTF-8"?> <file:Manifest xmlns:file="http://ns.osta.org/manifest/1.0/" xmlns:mpv="http://ns.osta.org/mpv/1.0/" xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/" xmlns:nmf="http://ns.osta.org/nmf/1.0/" > ... </file:Manifest>

		`--AaB03x--`
Explanation		• The **Content-Type** line indicates the type of content that follows, which is form data, url-encoded. This • **MPV_MANIFEST_LOCATION** is a form data variable whose value follows (in this case, "file:///E:/ALBUM.PVM" • **MPV_MANIFEST_ROOT_PATH** is a form data variable whose value is "file://E:/" • **MPV_UPLOAD_MANIFEST_MIME_**TYPE is a form data variable whose value is **"application/vnd.osta-org.upload.mpv+xml"** • **MPV_UPLOAD_RQST_FILE_EXT** is a form data variable whose value is **"mpvumpv"** • **MPV_MANIFEST_CONTENT** is the manifest content. • **MPV_INTERNET_PROFILE_VER**is "1.0"

HTML Example		`<!—Context-Aware Client-to-MPV Internet Profile provider Manifest Upload -->`
		`<HTML>`
		`  <HEAD>`
		`   <SCRIPT language="JavaScript">`
		
		`    // the MPV Manifest encoded as a JavaScript string`
		`    function get_Manifest()`
		`    {`
		`     var xml;`
		`     xml="<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";`
		`     xml+="<file:Manifest\n";`
		`     xml+="xmlns:file=\"http://ns.osta.org/manifest/1.0/\"\n";`
		`     xml+="xmlns:file=\"http://ns.osta.org/mpv/1.0/\"\n";`
		`     xml+="xmlns:file=\"http://ns.osta.org/presentation/1.0/\"\n";`
		`     xml+="xmlns:file=\"http://ns.osta.org/nmf/1.0/\">\n";`
		`     ... lots more here ...`
		`     xml+="</file:Manifest>\n";`
		`     return xml;`
		`    }`
		
		`    function get_ManifestLoc()`
		`    {`
		`       var loc;`
		`     loc = get_ManifestRoot();`
		`     loc += "ALBUM.PVM";`
		`     return loc;`
		`    }`
		
		`    function get_ManifestRoot()`
		`    {`
		`       var loc;`
		`     loc = window.location;`
		`     loc = loc.replace ("/\(.*\)\/.*(\?.*)?/\1\//"); //remove fname and all after q-mark`
		`     return loc;`
		`    }`
		
		`    function upload_Manifest()`
		`    {`
		`       // we know that the album file is in the same directory as the html page`
		`     form1.MPV_MANIFEST_LOCATION.value=get_ManifestLoc();`
		`     form1.MPV_MANIFEST_ROOT_PATH.value=get_ManifestRoot();`
		`     form1.MPV_MANIFEST_CONTENT.value=get_Manifest();`
		`     form1.action="http://www.somephotoservicesiteco.com/mpv/";`
		`     form1.submit()`

```
    }
   </SCRIPT>

  </HEAD>
  <BODY>
     <FORM ID="form1" NAME="form1"
           ENCTYPE="multipart/form-data"
           METHOD=POST>
          <A HREF="javascript:upload_Manifest()">Order Prints Online</A>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_CONTENT
NAME=MPV_MANIFEST_CONTENT>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_LOCATION
NAME=MPV_MANIFEST_LOCATION>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_ROOT_PATH
NAME=MPV_MANIFEST_ROOT_PATH>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_MANIFEST_MIME_TYPE
NAME=MPV_UPLOAD_MANIFEST_MIME_TYPE VALUE=”application/vnd.osta-
org.upload.mpv+xml”>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_RQST_FILE_EXT
NAME=MPV_UPLOAD_RQST_FILE_EXT VALUE=”mpvumpv”>
          <INPUT TYPE=HIDDEN
ID=MPV_INTERNET_PROFILE_VERNAME=MPV_INTERNET_PROFILE_VERVALUE="1.0">
     </FORM>
  </BODY>
</HTML>
```

Manifest Upload Encoded as application/x-www-form-urlencoded

The older protocol for form data upload based on HTTP POST and the “application/x-www-form-urlencoded” encoding MAY be supported. This encoding is more verbose for upload of large amounts of content.

SPEC	application/x-www-url-encoded Manifest Upload
Introduction	• Form-based Manifest Upload MUST utilize a HTTP POST request. • Form data is encoded using the application/x-www-form-urlencoded type • The following information MUST be provided: o MPV Manifest o Root Path to use for relative pathnames found in the manifest • The following information SHOULD be provided: o File-system location of the Manifest o MPV Internet Profile specification version of the upload protocol • The following information MAY be provided: o Mime type of the Asset Upload Request Manifest document. The default value is specified in Section 0, Asset Upload Client Identification. o File extension of the Asset Upload Request Manifest document. The default value is specified in Section 0, Asset Upload Client Identification.
HTTP Protocol (Requestor to MPV Internet Profile Provider) - POST protocol	**POST /{request-URI} HTTP/1.{0|1}** **{other HTTP request content}** **Content-Type: application/x-www-form-urlencoded** **MPV_MANIFEST_LOCATION={manifest-file-path}&MPV_MANIFEST_ROOT_PATH={root-path}&MPV_MANIFEST_CONTENT={manifest-**

for uploading a manifest as form variables	**content}&MPV_INTERNET_PROFILE_VER={ver} &MPV_UPLOAD_MANIFEST_MIME_TYPE={upload-manifest-mime-type}& MPV_SELECTED={selected-asset- id}&MPV_UPLOAD_RQST_FILE_EXT={upload-rqst-file-ext}**
Parameters	• **{request-URI}** is able to accept Conversation Requests with Form-based Manifest Upload • **MPV_MANIFEST_LOCATION** is the name of the field specifying a root-relative fully qualified path to the manifest file location or other suitable identifier. The MPV_MANIFEST_LOCATION value is passed back to the requestor by the MPV Internet Profile provider as the <mpv:LastURL> element of the <mpv:ManifestLink> in the Asset Upload Request. If no manifest location provided, asset upload is not possible because the upload client cannot access the manifest that contain the MPV assets referenced in the upload request. • **{manifest-file-path}** is the value of **MPV_MANIFEST_LOCATION** field. . In most cases, this will be a root-relative fully qualified path to the manifest file location. Exceptionally, it may be an arbitrary identifier that allows the client to identify the manifest; this might be appropriate for manifests stored in memory and never written to file, for example. It is a URL-encoded string. • **MPV_MANIFEST_ROOT_PATH** is the root path that is used with relative pathnames in the manifest to access client-side assets. • **{root-path}** is the value of **MPV_MANIFEST_ROOT_PATH** field. The path terminates in a slash "/". It may use any transport (e.g. file, http). It is a URL-encoded string. • **MPV_UPLOAD_MANIFEST_MIME_TYPE** is the field used to specify the mime type of the upload request manifest document. • **{upload-manifest-mime-type}** is the value **of MPV_UPLOAD_MANIFEST_MIME_TYPE** field. This string is used as the type of the upload request manifest document. • **MPV_UPLOAD_RQST_FILE_EXT** is the field used to specify the file extension of the upload request manifest document. • **{upload-rqst-file-ext}** is the value **of MPV_UPLOAD_RQST_FILE_EXT** field. This string is used as the file extension of the upload request manifest document. • **MPV_SELECTED** is the field used to specify an ID of a MPV asset that is pre-selected. More than one occurance of MPV_SELECTED fields may be present, allowing multiple assets to be pre-selected. If no assets are preselected, they are all preselected. Specifying a selected asset via a form variable is an alternative to specifying a selected asset via the <mpvb:MarkedAssets> element. • **{selected-asset-id}** is the value of the **MPV_SELECTED** field. This string must correspond to the asset id of an asset in the manifest content document. • **MPV_MANIFEST_CONTENT** is the name of the field specifying a valid MPV manifest as a string. • **{manifest-content}** is the value of **MPV_MANIFEST_CONTENT** field. It is URL-encoded a string. • **MPV_INTERNET_PROFILE_VER**is the name of the field specifying the MPV Internet Profile version number implemented.. • **{ver}** is "1.0"

EXAMPLE

The following example illustrates uploading the MPV manifest and related information to the MPV Internet Profile provider.

EXAMPLE	**application/x-www-url-encoded Manifest Upload**

HTTP Protocol (Requestor to MPV Internet Profile Provider)	`POST /mpv/ HTTP/1.1`
	`Accept: text/*, image/gif, image/jpg, */*`
	`Accept-Encoding: gzip`
	`Accept-Language: en, en_US`
	`Connection: Keep-Alive`
	`Cookie: userID=id456578`
	`Host: www.somephotoservicessite.com`
	`Referer: http://www.somephotoservicessite.com/index.html`
	`User-Agent: Mozilla/4.7 [en] (Win98; U)`
	`Content-Type: application/x-www-form-urlencoded`
	`Content-Length: 2349`
	
	`MPV_MANIFEST_LOCATION=file%3A%2F%2F%2FE%3A%47ALBUM.PVM&MPV_MANIFEST_ROOT_PATH=file%3A`
	`%2F%2F%2FE%3A%2F&MPV_MANIFEST_CONTENT=%3C%63xml%20version%3D%221%2E0%22%20encoding%3D%22`
	`-8%22?%3E%3Cfile%3AManifest%20xmlns%3D%22 ... %22%3E ...`
	`%3C%2Ffile%3AManifest%3E&MPVInternetProfileVER=%221%2E0%22&MPV_UPLOAD_MANIFEST_MIME_TYPE`
	`plication/vnd.osta-org.upload.mpv+xml&MPV_UPLOAD_RQST_FILE_EXT=mpvumpv`

Explanation	• The **Content-Type** line indicates the type of content that follows, which is form data, url-encoded. Thi • **MPV_MANIFEST_LOCATION** is a form data variable whose value follows (in this case, "file:///E:/ALBUM.PVM" • **MPV_MANIFEST_ROOT_PATH** is a form data variable whose value is "file://E:/" • **MPV_UPLOAD_MANIFEST_MIME_TYPE** is a form data variable whose value is **"application/vnd.osta-org.upload.mpv+xml"** • **MPV_UPLOAD_RQST_FILE_EXT** is a form data variable whose value is **"mpvumpv"** • **MPV_MANIFEST_CONTENT** is url-encoded string version of the manifest • **MPV_INTERNET_PROFILE_VER** is "1.0"

HTML Example	`<HTML>`
	`  <HEAD>`
	`   <SCRIPT language="JavaScript">`
	
	`    // the MPV Manifest encoded as a JavaScript string`
	`    function get_Manifest()`
	`    {`
	`     var xml;`
	`     xml="<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";`
	`     xml+="<file:Manifest\n";`
	`     xml+="xmlns:file=\"http://ns.osta.org/manifest/1.0/\"\n";`
	`     xml+="xmlns:file=\"http://ns.osta.org/mpv/1.0/\"\n";`
	`     xml+="xmlns:file=\"http://ns.osta.org/presentation/1.0/\"\n";`
	`     xml+="xmlns:file=\"http://ns.osta.org/nmf/1.0/\">\n";`
	`     ... lots more here ...`
	`     xml+="</file:Manifest>\n";`
	`     return xml;`
	`    }`
	
	`    function get_ManifestLoc()`
	`    {`
	`       var loc;`
	`     loc = get_ManifestRoot();`
	`     loc += "ALBUM.PVM";`
	`     return loc;`
	`    }`
	
	`    function get_ManifestRoot()`
	`    {`

```
        var loc;
      loc = window.location;
      loc = loc.replace ("/\(.*\)\/.*(\?.*)?/\1\//"); //remove fname and all after q-mark
      return loc;
    }

    function upload_Manifest()
    {
       // we know that the album file is in the same directory as the html page
      form1.MPV_MANIFEST_LOCATION.value=get_ManifestLoc();
      form1.MPV_MANIFEST_ROOT_PATH.value=get_ManifestRoot();
      form1.MPV_MANIFEST_CONTENT.value=get_Manifest();
      form1.action="http://www.somephotoservicesiteco.com/mpv/";
      form1.submit()
    }
   </SCRIPT>

  </HEAD>
  <BODY>
     <FORM ID="form1" NAME="form1"
           METHOD=POST>
          <A HREF="javascript:upload_Manifest()">Order Prints Online</A>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_LOCATION NAME=MPV_MANIFEST_LOCATION>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_CONTENT NAME=MPV_MANIFEST_CONTENT>
          <INPUT TYPE=HIDDEN ID=MPV_MANIFEST_ROOT_PATH NAME=MPV_MANIFEST_ROOT_PATH>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_MANIFEST_MIME_TYPE
NAME=MPV_UPLOAD_MANIFEST_MIME_TYPE VALUE=”application/vnd.osta-org.upload.mpv+xml”>
          <INPUT TYPE=HIDDEN ID=MPV_UPLOAD_RQST_FILE_EXT NAME=MPV_UPLOAD_RQST_FILE_EXT
VALUE=”mpvumpv”>
          <INPUT TYPE=HIDDEN ID=
MPV_INTERNET_PROFILE_VERNAME=MPV_INTERNET_PROFILE_VERVALUE="1.0">
     </FORM>
  </BODY>
</HTML>
```

Asset Transfer

The upload of images between a service requestor and a service provider is a critical part of the MPV Internet Profile-compliant interaction. For a MPV Internet Profile service, a client application interacts with the MPV Internet Profile provider and asset upload server. The basic architecture of asset transfer interaction is as follows

MPV Internet Profile provider initiates asset transfer by downloading an "asset upload request manifest" that identifyies which assets to transfer and to which destination (e.g. an asset upload server). A client-side uploader application processes the upload request.

The downloaded manifest specifies the following:

- The manifest to use to locate the client-side assets. Assets may NOT be specified directly by name; instead, a client-side manifest MUST be used to locate the assets. This is to avoid a security risk of allowing the server to specify the file to upload directly.
- A client-side root path to use with the manifest is NOT specified. All assets must be locatable from the client-side manifest using the search path mechanisms established for MPV. This is to avoid a security risk of allowing the server to specify the path to the file to upload
- A ContentID of each asset to upload MUST be provided in the upload manifest. A MPV Internet Profile-compliant upload client MUST decline to upload assets whose ContentID is invalid. This is to avoid a security risk of allowing the server to specify assets in a well-known manifest on the client system.
- The target destination for each client-side asset to be uploaded, along with any information about the lease on that URL.

Client-side Processing:

- Only those assets which are specified for upload are uploaded. Each asset is uploaded one at a time.

It is up to the client application to push the selected assets to the target URLs. Failure to upload the assets to these URLs before the time allowed in the lease(s) may result in a rejected order or other consequences.

It is left as an implementation detail of the service as to how the MPV Internet Profile service monitors the on-going status of the upload or is notified that the asset upload has completed. It may be the case that implementations decide to have a script that handles the HTTP POST that notifies the service after it has completed. Other implementations may choose to idly wait for the lease on the URLs to expire to determine if the asset has been uploaded. Any problem that may arise with this method and the solution is again left as an implementation detail.

Upload Request Manifest Security

It is very important to maintain the privacy of a user's MPV manifest. The manifest contains information about assets on the user's system, and servers with this information can initiate uploads of the user's assets, possibly without their awareness. This means that the manifest information should always be treated in a secure manner when transmitted over the internet. Therefore, a MPV Internet Profile client and server MUST always communicate over https **[SSL]** connections whenever upload request manifest information is present in the communication.

Thus the MPV Internet Profile upload request manifest MUST be transferred over a https connection.

Asset Upload Request

The Asset Upload Request is specified as a manifest that identifies which assets to upload.

SPEC	Asset Upload Request Manifest
Introduction	• The asset upload request manifest conforms to MPV Basic specification **[MPVBasic]** • Only assets that are identified in client-side local file-system manifests are uploaded. • The assets to upload are identified by reference within the "mpvb:MarkedAssets" element and the "mpv:MarkList" with the mpv:markType of "selected". • For each referenced asset in the MarkList, the associated LastURL properties provided in the nmf:Metadata are used as the target URL for upload. o The LeaseID, LeaseDur, and LeaseExpiresDate properties all relate to use of the target URL o Only one LastURL is used as the upload target. When multiple target LastURLs are present, one is selected by an unspecified method and used for the upload target and the other LastURLs are ignored. Good practice is to only provide one target URL o Root Path to use for relative pathnames found in the manifest • For each referenced asset in the MarkList, the associated ContentID value of the asset is provided. • When the upload is complete, the client GETs a completion URL supplied by the server • Only MPV simple assets may be uploaded; to upload MPV composite assets, decompose the composite asset into its constituent simple assets for uploading.
MIME type	The content MIME type of the Asset Upload Request Manifest may be specified during asset manifest upload using MPV_UPLOAD_MANIFEST_MIME_TYPE. The recommended distinguished MIME type for this Asset Upload Request Manifest is: **application/vnd.osta-org.upload.mpv+xml**
File extension	The file extension to use in the URL that requests the Asset Upload Request Manifest may be specified during asset manifest upload using MPV_UPLOAD_RQST_FILE_EXT. Only the first 8 characters of the extension string are used. To avoid collisions, the recommended pattern is "**mpvu[xxxx]**", where [xxxx] is a practically unique 4 letter sequence. The recommended distinguished file extension for this Asset Upload Request Manifest is: **mpvumpv**.
MPV profile	The MPV Internet Profile interface makes use of the MPV manifest with a specific set of required practices and schema elements. This usage MUST be recorded in the upload request manifest as a Profile in the ManifestProperties. For MPV Internet Profile 1.0, the profile URI is "http://www.osta.org/mpv/internet/1.0/".
XML Schema	With the exception of the following, the XML Schema for the asset upload request manifest is already fully specified by the **[MPVBasic]** specification. By extending the mpv:ManifestChildType, the mpvi:Context element is specified to occur as a top-level element of the MPV manifest. `<?xml version="1.0" encoding="UTF-8"?>` `<xs:schema targetNamespace="http://ns.osta.org/mpv/internet/1.0/" xmlns:mpv="http://ns.osta.org/mpv/1.0/" xmlns:mpvi="http://ns.osta.org/mpv/internet/1.0/" xmlns:nmf="http://ns.osta.org/nmf/1.0/" xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="qualified">` ` <xs:import namespace="http://ns.osta.org/mpv/1.0/" schemaLocation="core.xsd"/>`  ` <xs:element name="Context" type="mpvi:ContextType" substitutionGroup="mpv:ManifestChildBase"/>` ` <xs:complexType name="ContextType">`

```
        <xs:complexContent>
            <xs:extension base="mpv:ManifestChildType">
                <xs:sequence>
                    <xs:element name="UploadCompleteURL" type="xs:anyURI"/>
                    <xs:element name="AccessPoint" type="xs:anyURI"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

MPV Asset Upload Request Manifest Template

The following template manifest illustrates the specified usage.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
        xmlns:file="http://ns.osta.org/manifest/1.0/"
        xmlns:mpv="http://ns.osta.org/mpv/1.0/"
        xmlns:mpvb="http://ns.osta.org/mpv/basic/1.0/"
        xmlns:nmf="http://ns.osta.org/nmf/1.0/"
        xmlns:id="http://ns.osta.org/mpv/1.0/ident/"
        xmlns:mpvi="http://ns.osta.org /mpv/internet/1.0/"
        xmlns:url="http://ns.osta.org/mpv/1.0/ident/lasturl/">
  <nmf:Metadata>
    <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
      <ProfileBag>
          <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
          <Profile>http://ns.osta.org/mpv/internet/1.0/</Profile>
      </ProfileBag>
    </ManifestProperties>
  </nmf:Metadata>

  <mpvi:Context>
       <mpvi:UploadCompleteURL>{upload-complete-url}</mpvi:UploadCompleteURL>
       <mpvi:AccessPoint>{access-point}</mpvi:AccessPoint>
  </mpvi:Context>

  <mpvb:MarkedAssets>
    <mpv:MarkList mpv:markType="selected">
       <mpv:StillRef mpv:idRef="{asset-id}"
mpv:manifestLinkIdRef="MAN000100">
         <nmf:Metadata>
           <id:IdentProperties>
              <id:ContentID>urn:osta-org:mpv:dsig:md5:all:{md5-content-
id}</id:ContentID>
              <id:LastURL>
                 <url:LastURLProperties>
                    <url:leaseDur>{lease-dur}</url:leaseDur>
                    <url:LeaseExpiresDate>{lease-expires-
date}</url:LeaseExpiresDate>
                    <url:LeaseID>{lease-id}</url:LeaseID>
                    <url:URL>{target-URL}</url:URL>
                 </url:LastURLProperties>
              </id:LastURL>
           </id:IdentProperties>
         </nmf:Metadata>
       </mpv:StillRef>
       {repeat for additional asset references}

    </mpv:MarkList>
  </mpvb:MarkedAssets>

  <mpv:AssetList>
```

	<mpv:ManifestLink mpv:id="MAN000100"> <mpv:LastURL>**{manifest-file}**</mpv:LastURL> </mpv:ManifestLink> </mpv:AssetList> </file:Manifest>
Parameters	• **{upload-complete-url}** is the URL the upload application should GET using HTTP 1.0 when upload of all the assets is complete. • **{asset-id}** is the ID string for the asset in the {manifest-file} • **{md5-content-id}** is a MD5 content hash of the asset • **{target-URL}** is the target URL for the asset • **{lease-id}** is the lease ID for the {target-URL} • **{lease-dur}** is the lease duration, in seconds from the time the manifest is provided, for the {target-URL} • **{lease-expires-date}** is the approximate date the {target-URL} lease expires. This is imprecise given possible variation in the local time of the client versus the server that issued the lease. • **{manifest-file}** is the name of the client-side manifest that contains the asset information. • **{additional-asset-references}** are additional reference to assets to upload that utilize the same template of the <mpv:StillRef> element and child elements again. Note that the asset type of the reference may be any of the known MPV simple asset types, such as <mpv:VideoRef> or <mpv:AudioRef>. Composite assets, such as <mpv:StillWithAudio> or <mpv:StillMultishotSequence> may not be used; instead, upload the simple assets contained by the composite asset.

The asset upload request manifest is sent from the MPV Internet Profile provider to the client-side asset upload application using the web browser as the intermediary. The MPV Internet Profile provider replies to a web browser GET request with the manifest document whose content type is set to application/vnd.osta-org.upload.mpv+xml. The web browser will hand off the manifest to the registered handler for this type.

SPEC	**Asset Upload Request Protocol**
Introduction	• Upload of the assets is requested by downloading an Asset Upload Request Manifest using the content-type of application/vnd.osta-org.upload.mpv+xml • The web browser activates the appropriate content type handler for the asset upload request. o Each web browser has its own mechanisms for handling typed-content. In a PC environment, a discrete upload application may be registered to handle this content type. In firmware, the content type may be handled within a custom browser or as a separate application.
HTTP Protocol (MPV Internet Profile provider to client)	HTTP/1.0 200 OK Date: Tue, 22 Jul 2002 23:59:59 GMT Content-Type: **{upload-manifest-mime-type}** Content-Length: **{size}** **{asset-upload-request-manifest}**
Parameters	• **{upload-manifest-mime-type}** is the MIME type of an asset upload request manifest, which will be handled by the appropriate client-side handler. **application/vnd.osta-org.upload.mpv+xml** is the recommended distinguished value. • **{size}** is the size of the content in accordance with HTTP • **{asset-upload-request-manifest}** is the asset upload request manifest as

	specified by this document.

MPV Internet Profile defines a distinguished recommended content type for the asset upload request manifest, **application/vnd.osta-org.upload.mpv+xml**. An implementation may also specify its own mime type for the asset upload request manifest as part of the manifest upload. MPV Internet Profile defines a distinguished recommended file extension for the asset upload request manifest, **mpvumpv**. An implementation may also specify its own file extension for the asset upload request manifest as part of the asset manifest upload.

SPEC	Asset Upload Request Manifest Identification
Introduction	• The content type of a URL is determined in a web browser-specific manner. Widely implemented techniques utilize either or both file extension and HTTP content mime type. • For example, on Microsoft Windows system with Microsoft Internet Explorer, an application can be registered to handle both file extensions and content mime types. In certain cases, primarily concerning asset upload client installation behaviour, the file extension association is the more important to achieve the desired user experience.
Asset Upload Request Manifest Document File Extension in Request URL	• The Asset Upload Request Manifest document URL is specified by the web server as part of the presentation content it provides. • The URL MUST use either of the following: o If specified in the manifest upload via MPV_UPLOAD_RQST_FILE_EXT, the URL MUST use the {upload-rqst-file-ext} extension on the filename o If no file extension was specified, the default extension is "**mpvumpv**". • Note that the file extension MUST NOT be the standard MPV extension (.pvm) because this is associated with the default MPV viewer application, not the asset upload client.
Asset Upload Request Manifest Document Content Mime type	• The Asset Upload Request Manifest document is provided by the web server in response to the request. • The Asset Upload Request Manifest document MUST use either of the following content types in its request response: o If specified in the manifest upload via MPV_UPLOAD_MANIFEST_MIME_TYPE, the URL MUST use the {upload-manifest-mime-type} mime type for the document content type o If no mime type was specified, the default mime type is "**application/vnd.osta-org.upload.mpv+xml**".

EXAMPLE	Asset Upload Client Identification
HTTP Protocol (Requestor to MPV Internet Profile Provider)	**GET /mpv/ HTTP/1.1** **Accept: text/*, image/gif, image/jpg, application/vnd.osta-org.upload.mpv+xml */*** **Accept-Encoding: gzip** **Accept-Language: en, en_US** **Connection: Keep-Alive** **Cookie: userID=id456578** **Host: www.somephotoservicessite.com** **Referer: http://www.somephotoservicessite.com/index.html** **User-Agent: Mozilla/4.7 [en] (Win98; U)**
Client-side script	**… to be provided …**

IMPLEMENTATION NOTE

The hand-off of control from the web browser to the asset upload client is system-specific. In some environments, such as Microsoft Windows systems utilizing Microsoft Internet Explorer, the web browser issues the GET request

for the Asset Upload Request Manifest but breaks off the download after detecting the content type in the header of the reply. It then delegates download of the Asset Upload Request Manifest to the asset upload client. This results in the server experiencing two GET requests for the Asset Upload Request Manifest, the first is partial and the second complete.

EXAMPLE

The client-side manifest is given in Chapter 0, Sample Data. The Asset Upload Request manifest provided by the MPV Internet Profile provider to initiate upload is as follows:

EXAMPLE	Asset Upload Request Manifest
Asset Upload Request MPV Manifest Example, **upload-manifest.mpvumpv**	

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
        xmlns:file="http://ns.osta.org/manifest/1.0/"
        xmlns:mpv="http://ns.osta.org/mpv/1.0/"
        xmlns:mpvb="http://ns.osta.org/mpv/basic/1.0/"
        xmlns:nmf="http://ns.osta.org/nmf/1.0/"
        xmlns:id="http://ns.osta.org/mpv/1.0/ident/"
        xmlns:url="http://ns.osta.org/mpv/1.0/ident/lasturl/">
  <nmf:Metadata>
    <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
      <ProfileBag>
          <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
      </ProfileBag>
    </ManifestProperties>
  </nmf:Metadata>

  <mpvi:Context>
       <mpvi:UploadCompleteURL> http://upload.somephotoservicessite.com/23493209-2345
2345-2346-23459BB98736</mpvi:UploadCompleteURL>
       <mpvi:AccessPoint>
https://upload.somephotoservicessite.com/mpv/core</mpvi:AccessPoint>
  </mpvi:Context>

  <mpvb:MarkedAssets>
    <mpv:MarkList mpv:markType="selected">
       <mpv:StillRef mpv:idRef="ID000100" mpv:manifestLinkIdRef="MAN000100">
         <nmf:Metadata>
           <id:IdentProperties>
              <id:ContentID>urn:osta-
org:mpv:dsig:md5:all:3F886AEFA3B340da971BAF09B17DBC122</id:ContentID>
              <id:LastURL>
                 <url:LastURLProperties>
                    <url:leaseDur>172800000</url:leaseDur> <!-- two days -->
                    <url:LeaseID>92BD4323-3462-2399-8934-BCC9-39929934</url:LeaseID>
                    <url:URL>http://upload.somephotoservicessite.com/23493209-2345-23
2346-2345-9BB98736</url:URL>
                 </url:LastURLProperties>
              </id:LastURL>
           </id:IdentProperties>
         </nmf:Metadata>
       </mpv:StillRef>
       <mpv:StillRef mpv:idRef="ID000200" mpv:manifestLinkIdRef="MAN000100">
         <nmf:Metadata>
           <id:IdentProperties>
              <id:ContentID>urn:osta-
org:mpv:dsig:md5:all:F9886AEFA3B340da971BAF09B17DBC199</id:ContentID>
              <id:LastURL>
                 <url:LastURLProperties>
                    <url:leaseDur>172800000</url:leaseDur> <!-- two days -->
```

```
                    <url:LeaseID>92BD4323-3462-2399-8934-BCC9-39929934</url:LeaseID>
                    <url:URL>http://upload.somephotoservicessite.com/2BBC3209-2345-23
2346-2345-9BB988723</url:URL>
                 </url:LastURLProperties>
              </id:LastURL>
           </id:IdentProperties>
         </nmf:Metadata>
       </mpv:StillRef>
       <mpv:StillRef mpv:idRef="ID000300" mpv:manifestLinkIdRef="MAN000100">
         <nmf:Metadata>
           <id:IdentProperties>
              <id:ContentID>urn:osta-
org:mpv:dsig:md5:all:23886AEFA3B340da971BAF09B17DBC161</id:ContentID>
              <id:LastURL>
                 <url:LastURLProperties>
                    <url:leaseDur>172800000</url:leaseDur> <!-- two days -->
                    <url:LeaseID>92BD4323-3462-2399-8934-BCC9-39929934</url:LeaseID>
                    <url:URL>http://upload.somephotoservicessite.com/99343209-2345-23
2346-2345-9BB99022</url:URL>
                 </url:LastURLProperties>
              </id:LastURL>
           </id:IdentProperties>
         </nmf:Metadata>
       </mpv:StillRef>
    </mpv:MarkList>
  </mpvb:MarkedAssets>

  <mpv:AssetList>

    <mpv:ManifestLink mpv:id="MAN000100">
      <mpv:LastURL>file:///E:/ALBUM.PVM</mpv:LastURL>
    </mpv:ManifestLink>

  </mpv:AssetList>

</file:Manifest>
```

EXAMPLE	**Asset Upload Request Protocol**
HTTP Protocol (MPV Internet Profile provider to client)	HTTP/1.0 200 OK Date: Tue, 22 Jul 2002 23:59:59 GMT Content-Type: application/vnd.osta-org.upload.mpv+xml Content-Length: 1394

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
        xmlns:file="http://ns.osta.org/manifest/1.0/"
        xmlns:mpv="http://ns.osta.org/mpv/1.0/"
        xmlns:mpvb="http://ns.osta.org/mpv/basic/1.0/"
        xmlns:nmf="http://ns.osta.org/nmf/1.0/"
        xmlns:id="http://ns.osta.org/mpv/1.0/ident/"
        xmlns:url="http://ns.osta.org/mpv/1.0/ident/lasturl/">
  <nmf:Metadata>
    <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
      <ProfileBag>
          <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
      </ProfileBag>
    </ManifestProperties>
  </nmf:Metadata>
```

```
  <mpvi:Context>
       <mpvi:UploadCompleteURL>
http://upload.somephotoservicessite.com/23493209-2345-2345-2346-
23459BB98736</mpvi:UploadCompleteURL>
       <mpvi:AccessPoint>
https://upload.somephotoservicessite.com/mpv/core</mpvi:AccessPoint>
  </mpvi:Context>

  <mpvb:MarkedAssets>
    <mpv:MarkList mpv:markType="selected">
       <mpv:StillRef mpv:idRef="ID000100"
mpv:manifestLinkIdRef="MAN000100">
         <nmf:Metadata>
           <id:IdentProperties>
              <id:ContentID>urn:osta-
org:mpv:dsig:md5:all:3F886AEFA3B340da971BAF09B17DBC122</id:ContentID>
              <id:LastURL>
                 <url:LastURLProperties>
                    <url:leaseDur>172800</url:leaseDur> <!-- two days -->
                    <url:LeaseID>92BD4323-3462-2399-8934-BCC9-
39929934</url:LeaseID>

<url:URL>http://upload.somephotoservicessite.com/23493209-2345-2345-2346-
2345-9BB98736</url:URL>
                 </url:LastURLProperties>
              </id:LastURL>
           </id:IdentProperties>
         </nmf:Metadata>
       </mpv:StillRef>
       <mpv:StillRef mpv:idRef="ID000200"
mpv:manifestLinkIdRef="MAN000100">
         <nmf:Metadata>
           <id:IdentProperties>
              <id:ContentID>urn:osta-
org:mpv:dsig:md5:all:F9886AEFA3B340da971BAF09B17DBC199</id:ContentID>
              <id:LastURL>
                 <url:LastURLProperties>
                    <url:leaseDur>172800</url:leaseDur> <!-- two days -->
                    <url:LeaseID>92BD4323-3462-2399-8934-BCC9-
39929934</url:LeaseID>

<url:URL>http://upload.somephotoservicessite.com/2BBC3209-2345-2345-2346-
2345-9BB988723</url:URL>
                 </url:LastURLProperties>
              </id:LastURL>
           </id:IdentProperties>
         </nmf:Metadata>
       </mpv:StillRef>
       <mpv:StillRef mpv:idRef="ID000300"
mpv:manifestLinkIdRef="MAN000100">
         <nmf:Metadata>
           <id:IdentProperties>
              <id:ContentID>urn:osta-
org:mpv:dsig:md5:all:23886AEFA3B340da971BAF09B17DBC161</id:ContentID>
              <id:LastURL>
                 <url:LastURLProperties>
                    <url:leaseDur>172800</url:leaseDur> <!-- two days -->
                    <url:LeaseID>92BD4323-3462-2399-8934-BCC9-
39929934</url:LeaseID>

<url:URL>http://upload.somephotoservicessite.com/99343209-2345-2345-2346-
2345-9BB99022</url:URL>
                 </url:LastURLProperties>
```

		</id:LastURL> </id:IdentProperties> </nmf:Metadata> </mpv:StillRef> </mpv:MarkList> </mpvb:MarkedAssets> <mpv:AssetList> <mpv:ManifestLink mpv:id="MAN000100"> <mpv:LastURL>file:///E:/ALBUM.PVM</mpv:LastURL> </mpv:ManifestLink> </mpv:AssetList> </file:Manifest>

EXAMPLE	**Asset Upload Request Implementation**
Ready to Upload Assets Web Page (MPV Internet Profile provider to web browser), **ready-to-upload.jsp**	<html lang="en"> <head> <title>Ready to Upload</title> <base href="http://localhost:8080/mpv/ready-to-upload.jsp"> </head> <body bgcolor="white"> Order Complete, Ready to Upload Images Upload images now </body> </html>
Asset Upload Web Page (MPV Internet Profile provider to web browser), **initiate-upload.jsp**	<html lang="en"> <head> <base href="http://localhost:8080/mpv/upload-frameset.jsp"> <title>upload frameset</title> </head> <frameset rows="*,0"> <frame noresize src="upload-page.jsp"> <frame noresize src="**upload-manifest.mpvumpv**"> </frameset> </html>
Asset Upload Request Manifest (MPV Internet Profile provider to upload client), **upload-manifest.mpvumpv**	See example upload manifest elsewhere in this section.
Asset Upload Request Manifest Upload Status Page	<html lang="en"> <!-- This example upload page monitors a server-provided status

(MPV Internet Profile provider to web browser), **upload-page.jsp**

```
document to provide the user with regularly updated upload status.
The server-client status protocol is entirely determined by the web
server implementation.  -->
  <head>
    <base href="http://localhost:8080/mpv/upload-page.jsp">
    <title>Image Upload</title>
    <script language="JavaScript">

        var document;

        function GetUploadStatus()
        {
            document.load("http://localhost:8080/mpv/upload-
status.do");
        }

        function OnReadyStateChange()
        {
            if(document.readyState == 4) // 4 == document completely
loaded
            {
                var currentThumbnailUrlNode =
document.selectSingleNode("//CurrentThumbnailUrl");
                if(currentThumbnailUrlNode != null)
                {
                     Thumbnails.innerHTML =
                        '<img src="' + currentThumbnailUrlNode.text +
'">';
                }

                var currentFileNumberNode =
document.selectSingleNode("//CurrentFileNumber");
                if(currentFileNumberNode != null)
                {
                     var currentFileNumber =
currentFileNumberNode.text;
                }
                var totalFilesNode =
document.selectSingleNode("//TotalFiles");
                if(totalFilesNode != null)
                {
                     var totalFiles = totalFilesNode.text;
                }
                UploadMessage.innerHTML =
                    "Uploading asset <b>" +
                    currentFileNumber + "</b> of <b>" +
                    totalFiles + "</b>...";

                var uploadCompleteNode =
document.selectSingleNode("//UploadComplete");
                if(uploadCompleteNode != null)
                {
                    var uploadComplete = uploadCompleteNode.text;
                    if(uploadComplete == "true")
                    {
                        UploadMessage.innerHTML = "Upload complete.";
                    }
                    else
                    {
                        GetUploadStatus();
                    }
                }
            }
```

```
        }

        function BodyOnLoad()
        {
            document = new ActiveXObject( "Microsoft.XMLDOM" );
            if(null == document)
            {
                alert("Missing Microsoft.XMLDOM. Unable to show
upload status.");
                return;
            }
          document.async = true;
          document.resolveExternals = false;
             document.onreadystatechange = OnReadyStateChange;
             GetUploadStatus();
        }

    </script>
  </head>

  <body onload="BodyOnLoad()">
    <h3>Upload Page</h3>
    <p></p>
    <p id="UploadMessage"></p>
    <p id="Thumbnails"></p>
  </body>

</html>
```

Asset Upload Protocol

Once the asset upload client application has the asset upload request manifest, it processes it and uploads the requested assets according to the following protocol.

SPEC	Asset Upload Protocol
Introduction	• Upload of the assets uses the HTTP POST protocol for file upload **[HTTP1.1][FileUpload]** • The MPV_INTERNET_PROFILE_VERfield value SHOULD be provided. The value is 1.0. • Each asset is uploaded as a discrete and separate POST message containing only the single file payload o The form data field name is ASSET o If the content-type is known, it MAY be provided; this is not required o If the filename is known, it MAY be provided; this is not required

HTTP Protocol (Requestor to MPV Internet Profile Provider)	`POST /{target-URL} HTTP/1.1`
	`{other HTTP content}`
	`Content-type: multipart/form-data, boundary={boundary identifier}`
	`Content-length: {size}`
	
	`--{boundary identifier}`
	`content-disposition: form-data; name=" MPV Internet Profile_VER"`
	
	`{ver}`
	`--{boundary identifier}`
	`content-disposition: form-data; name="ASSET1"; filename="foo.jpg"`
	`content-type: image/jpeg`

	{asset data} --**{boundary identifier}**--
Parameters	• **{target-URL}** is the target URL for the asset • **multipart/form-data** is the content type in accordance to HTTP. • **{boundary-identifier}** is generated by the client in accordance to HTTP. • **{size}** is generated by the client in accordance to HTTP. • **ASSET** is the name of the field containing the asset data • **{asset data}** is the data of the asset being uploaded • **MPV_INTERNET_PROFILE_VER**is the name of the field specifying the MPV Internet Profile version number implemented. • **{ver}** is "1.0"
Protocol	The Asset Upload client application must support upload using the http and https protocols. This means that the {target-URL} may use either http or https URLs.

EXAMPLE	**Asset Upload Protocol**
HTTP Protocol (Requestor to MPV Internet Profile Provider)	POST **/99343209-2345-2345-2346-2345-9BB99022** HTTP/1.1 {other HTTP content} **Host: www.somephotoservicessite.com** User-Agent: MPV Internet Profile-Upload1.0 Host: upload.somephotoservicessite.com Content-type: **multipart/form-data**, boundary=**Abzef9** Content-length: 983423 --**Abzef9** content-disposition: form-data; name="**MPV_Internet Profile_VER**" **1.0** --**Abzef9** content-disposition: form-data; name="**ASSET**"; filename="foo.jpg" content-type: image/jpeg **... foo.jpg data ...** --**Abzef9**--
Parameters	• **/99343209-2345-2345-2346-2345-9BB99022** is the target URL for the asset • **multipart/form-data** is the content type in accordance to HTTP. • **Abzef9** is the boundary identifier in accordance to HTTP. • **ASSET** is the name of the field containing the asset data • **foo.jpg data** is the data of the asset being uploaded • **MPV_INTERNET_PROFILE_VER**is the name of the field specifying the MPV Internet Profile version number implemented, which is "1.0"

Asset Upload Client Identification

MPV Internet Profile defines a distinguished recommended content type for the asset upload request manifest, **application/vnd.osta-org.upload.mpv+xml**. An implementation may also specify its own mime type for the asset upload request manifest as part of the manifest upload. A web browser that is able to handle this content type may report this fact as part of its conversation with the MPV Internet Profile provider using standard HTTP conventions **[HTTP1.1]**. If not reported in the conversation request, client-side scripting in the web browser can typically determine whether a browser supports a given content type.

The Asset Upload Client is installed on the client system in a system- and client-specific manner. On a typical PC with a modern Microsoft Windows operating system and Microsoft Internet Explorer, for example, an application can be registered to handle documents of a given mime type or file extension. This application is invoked by the web browser when a document of that type is requested to be opened.

SPEC	Asset Upload Client Identification
Asset Upload Request Manifest Document Request URL	• The Asset Upload Request Manifest document URL is specified by the web server as part of the presentation content it provides. • The URL MUST use either of the following: o If specified in the manifest upload via MPV_UPLOAD_RQST_FILE_EXT, the URL MUST use the {upload-rqst-file-ext} extension on the filename o If no file extension was specified, the default extension is "**mpvumpv**".
Content types supported by Client	• The Client MAY specify which content types it supports as part of the HTTP request using the Accept header. o It is recommended for the client to announce support for the mime type that is to be used by the server for the Asset Upload Request Manifest document. o If specified in the manifest upload via MPV_UPLOAD_MANIFEST_MIME_TYPE, the the {upload-manifest-mime-type} mime type SHOULD be used for the document content type o If no mime type was specified, the recommend mime type is "**application/vnd.osta-org.upload.mpv+xml**".
Asset Upload Request Manifest Document Content Mime type	• The Asset Upload Request Manifest document is provided by the web server in response to the request. • The Asset Upload Request Manifest document MUST use either of the following content types in its request response: o If specified in the manifest upload via MPV_UPLOAD_MANIFEST_MIME_TYPE, the URL MUST use the {upload-manifest-mime-type} mime type for the document content type o If no mime type was specified, the default mime type is "**application/vnd.osta-org.upload.mpv+xml**".

EXAMPLE	Asset Upload Client Identification
HTTP Protocol (Requestor to MPV Internet Profile Provider)	**GET /mpv/ HTTP/1.1** **Accept: text/*, image/gif, image/jpg, application/vnd.osta-org.upload.mpv+xml */*** **Accept-Encoding: gzip** **Accept-Language: en, en_US** **Connection: Keep-Alive** **Cookie: userID=id456578** **Host: www.somephotoservicessite.com** **Referer: http://www.somephotoservicessite.com/index.html** **User-Agent: Mozilla/4.7 [en] (Win98; U)**
Client-side script	**… to be provided …**

Asset Upload Security Risks and Mitigation

The ability of the server to request upload of specific assets from a client system is a security risk. It is potentially hard to detect security violations because the upload client may choose not to present any user interface and may not allow user intervention. The risks to consider are as follows:

The principle concerns addressed by this specification are two-fold. First, the server is prevented from explicitly specifying the path of the file to upload. This forces a malicious server to make use of manifests already on the client system, and to know their location explicitly.

Secondly, to server is required to demonstrate explicit knowledge about the assets to upload that is difficult to spoof. This is achieved by requiring the server-produced upload manifest to specify the ID of each asset to upload, and most importantly, to specify a ContentID of each asset to upload. The ContentID value in the local manifest must match the ContentID value supplied Asset Upload Request Manifest. However, it is not required that the actual ContentID of asset match – this prevents the upload client from having to know about all of the ContentID computation methods which might be used.

Some upload clients may choose to implement a user interface that announce when they are being used. This could aid in detection of malicious servers.

Typographic Conventions

Chapter 4: MPV Internet Profile Practices

Appendix I: References

[DATETIME]

"Date and Time Formats", M. Wolf, C. Wicksteed. W3C Note 27 August 1998, Available at: http://www.w3.org/TR/NOTE-datetime

[DC]

"Dublin Core Metadata Initiative", a Simple Content Description Model for Electronic Resources. Available at http://purl.org/DC/

[DC-NMF]

"Dublin Core Normalized Metadata Format Profile Specification 1.0"; OSTA, 2002. Available at http://www.osta.org/mpv/

[DCF-1999]

“Design rule for Camera File system, Version 1.0”, JEIDA standard, English Version 1999.1.7, Japanese Electronic Industry Development Association (JEIDA).

[DIG35-2001]

“DIG35 Specification – Metadata for Digital Images, Version 1.1”, June 18, 2001, International Imaging Industry Association (I3A) [recently formed by combining the Digital Imaging Group and PIMA]. http://www.i3a.org

[ISO8601]

"Data elements and interchange formats - Information interchange - Representation of dates and times", International Organization for Standardization, 1998.

[ISO10646]

""Information Technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-1:1993. This reference refers to a set of codepoints that may evolve as new characters are assigned to them. This reference therefore includes future amendments as long as they do not change character assignments up to and including the first five amendments to ISO/IEC 10646-1:1993. Also, this reference assumes that the character sets defined by ISO 10646 and Unicode remain character-by-character equivalent. This reference also includes future publications of other parts of 10646 (i.e., other than Part 1) that define characters in planes 1-16. "

[JFIF]

"JPEG File Interchange Format, Version 1.02"; Eric Hamilton, September 1992. Available at http://www.w3.org/Graphics/JPEG/jfif.txt

[MANIFEST]

"XML Manifest Specification 1.0"; OSTA, 2002,.
Available at http://www.osta.org/mpv/

[MD5]

"The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
Available at http://www.ietf.org/rfc/rfc1321.txt. Further information and source code available at http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html

[MIME-2]

"RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types"; N. Freed, N. Borenstein, November 1996.
Available at ftp://ftp.isi.edu/in-notes/rfc2046.txt

[MIMETYPES-REG]

IANA official registry of MIME media types
Available at http://www.isi.edu/in-notes/iana/assignments/media-types/media-types

[MPVBasic]

"MPV – Basic Profile Specification", OSTA, 2002,
Available at http://www.osta.org/mpv/

[MPVCore]

"MPV Core Specification 1.0"; OSTA, 2002,.
Available at http://www.osta.org/mpv/

[MPVDoc]

"MPV Internet Profile Specification 1.0"; OSTA, 2003,.
Available at http://www.osta.org/

[MPVPres]

"MPV Presentation Profile Specification 1.0"; OSTA, 2002,.
Available at http://www.osta.org/mpv/

[NMF]

"Normalized Metadata Format Specification 1.0"; OSTA, 2002,.
Available at http://www.osta.org/mpv/

[PNG-MIME]

"Registration of new Media Type image/png"; Glenn Randers-Pehrson, Thomas Boutell, 27 July 1996.
Available at ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/image/png

[PNG-REC]

"PNG (Portable Network Graphics) Specification Version 1.0"; Thomas Boutell (Ed.).
Available at http://www.w3.org/TR/REC-png

[QT]

"QuickTime Movie File Format Specification", May 1996.
Available at http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/refFileFormat96.htm

[QT-MIME]

"Registration of new MIME content-type/subtype"; Paul Lindner, 1993.
Available at http://www.isi.edu/in-notes/iana/assignments/media-types/video/quicktime

[RDFsyntax]

"Resource Description Framework (RDF) Model and Syntax Specification", Ora Lassila and Ralph R. Swick. W3C Recommendation 22 February 1999,
Available at http://www.w3.org/TR/REC-rdf-syntax/

[RDFschema]

"Resource Description Framework (RDF) Schema Specification", Dan Brickley and R.V. Guha.W3C Proposed Recommendation 03 March 1999,
Available at http://www.w3.org/TR/PR-rdf-schema/

[RFC1766]

"Tags for the Identification of Languages", H. Alvestrand, March 1995.
Available at ftp://ftp.isi.edu/in-notes/rfc1766.txt

[RFC2119]

“Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, IETF RFC 2119
http://www.ietf.org/rfc/rfc2119.txt

[URI]

"Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, August 1998. Note that RFC 2396 updates [RFC1738] and [RFC1808].

[UCS-2]

16-bit encoding of ISO 10646, commonly known as the Unicode character set.

[UTF-8]

Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[W3C-NSURI]

"URIs for W3C namespaces". Policy and administrative issue for W3C, Oct. 1999.
Available at http://www.w3.org/1999/10/nsuri

[XML10]

"Extensible Markup Language (XML) 1.0" T. Bray, J. Paoli and C.M. Sperberg-McQueen. W3C Recommendation 10 February 1998 ,
Available at http://www.w3.org/TR/REC-xml

[XML-NS]

"Namespaces in XML", Tim Bray, Dave Hollander, Andrew Layman. W3C Recommendation 14 January 1999,
Available at http://www.w3.org/TR/REC-xml-names

[XSCHEMA]

"XML Schema, XML Schema Part 1: Structures". W3C Working Draft, work in progress.
Available at http://www.w3.org/TR/xmlschema-1/

[XSL]

"Extensible Stylesheet Language (XSL) Specification", Stephen Deach. W3C Working Draft, work in progress.
Available at http://www.w3.org/TR/xsl/