



API Reference

Created on 06/20/2009

**Contact Estate⁺⁺ to obtain your free Vendor API key.
support@estateplusplus.com**

Table of Contents

IMPORTANT TERMINOLOGY3
SLICES OF INFORMATION.....4
REAL LIFE SCENARIOS.....5
BEFORE USING THE APIs5
UPLOADBYFOLDERNAME6
UPLOADBYFILESYSTEMPATH.....7
ERROR CODES8

Important Terminology

An **Estate** is an entity within Estate⁺⁺ that represents the compilation of all data entered for an individual household. This includes uploaded documents and all manually entered information. A **Partner Account** is an entity within Estate⁺⁺ that represents a Corporate Partner. Partner accounts have limited role-based file sharing abilities with an estate. An **Individual Account** is an entity in Estate⁺⁺ that represents a single estate and its associated entities, such as shared users (full access and read only), partner associations, folders, documents, and data entered. An individual account is created when an end-user registers with Estate⁺⁺.

As a software vendor you will be most interested in **documents and folders**. The APIs exposed will allow you to upload documents created by your software to folders associated with an estate. Due to the restrictive nature of Estate⁺⁺, an understanding of the folders associated with an estate is necessary before using the API.

Private folders can only be accessed through the website by Estate⁺⁺ users who have full or read only authority. A full access user can create and delete private folders. When an estate is created, a set of private folders are also created called *Root*, *Medical*, *Legal*, and *Family*. These folders cannot be accessed by any of the APIs.

A **Partner Dropbox folder** is a special shared folder that allows two-way file sharing between an estate and a designated corporate partner such as an estate planning attorney, a tax accountant, or a financial advisor. When an estate is created, a set of partner dropbox folders are created called *Legal Dropbox*, *Tax Dropbox*, and *Financial Dropbox*. These folders can be accessed through the website by Estate⁺⁺ users who have full or read only authority. These folders cannot be deleted. These folders cannot be accessed by any of the APIs.

The **Local Dropbox folder** is a special shared folder that allows file sharing and synchronization between an estate and a designated folder located on a personal computer. This allows a user to upload multiple files without having to use the website. A user agent running on the personal computer performs synchronization as a task running in the background. The user agent can be installed on more than one computer. Files are synchronized between all computers running the user agent. Full access and read only users can access this folder through the website. This folder cannot be accessed by any of the APIs.

Vendor specific folders are created by the UploadByFolderName API. Access to these folders is restricted to a single Vendor API key and the UploadByFolderName API. This prevents multiple vendors from storing (and possibly overwriting) files in the same folder. Full access and read only users can access these folders through the website. These folders cannot be accessed by any other API.

File system folders are created by the UploadByFileSystemPath API. Although a Vendor API key is required to use this API, access to these folders is only restricted to the API itself. The UploadByFileSystemPath API tries to mimic the local file system, effectively allowing files of the same name to be overwritten in the same folder by multiple vendors. Full access and read only users can access these folders through the website. These folders cannot be accessed by any other API.

Each estate has a public and private key, known collectively as **Access Identifiers**. The public key identifies a specific estate to the API. The private key is used to produce a request signature to prevent unauthorized of the HTTP request and a digital signature to prevent tampering of the file being uploaded. Users can obtain a public and private key by logging into Estate⁺⁺ and navigating to: *Account Management-->Access Identifiers*

Slices of Information

The Estate⁺⁺ collaboration features allow information to be shared on a "need to know" basis as independent slices of information. For example, a tax accountant needs access to tax returns. An attorney needs access to legal documents. An executor of an estate needs access to wills and trusts. A physician may need access to advanced medical directives. Key family members may need access to all information.

A single full-access user is created at the same time an estate is created. A full-access user can add, delete, or change any information or documents associated with an estate. This user can create many different entities, each with selective access to information, including:

- **Additional full-access users**, usually for a spouse or domestic partner.
- **Read-only users** who can retrieve information as a single PDF file and download any document. Read-only users are typically key family members and the executor of an estate.
- **Read-only users** who can only download **medical directives**. A health care professional must first register with Estate⁺⁺ before they can be associated with an estate's medical directives.
- Two-way file sharing between an estate and a designated advisor such as an estate planning attorney using the **Legal Dropbox**. A legal advisor must first register with Estate⁺⁺ before they can be associated with an estate's Legal Dropbox.
- Two-way file sharing between an estate and a designated advisor such as a tax accountant using the **Tax Dropbox**. A tax advisor must first register with Estate⁺⁺ before they can be associated with an estate's Tax Dropbox.
- Two-way file sharing between an estate and a designated advisor such as a financial planner using the **Financial Dropbox**. A financial advisor must first register with Estate⁺⁺ before they can be associated with an estate's Financial Dropbox.
- File sharing and synchronization between an estate and multiple computers using the **Local Dropbox**. The local dropbox user agent must be installed on each computer.
- **Direct links** to specific documents located within an estate. Each direct link can be used to download a single document. Direct links are available for any document in any folder. Direct links can only be generated by full access users. For safety purposes, direct links expire within 1 hour, 4 hours, and 24 hours. Direct links also expire immediately when a full access user generates a new access identifier private key.
- File uploading between an estate and a software product using the **UploadByFolderName API**. This API allows unique access to folders dedicated to a *single* software vendor. Software vendors must be registered with Estate⁺⁺ before they can use the API in a production environment.
- File uploading between an estate and a software product using the **UploadByFileSystemPath API**. This API seeks to mimic the underlying local file system and allows access to folders by *multiple* software vendors. Software vendors must be registered with Estate⁺⁺ before they can use the API in a production environment.

Real Life Scenarios

Scenario: As the head of household, you enter information about your passport, credit cards, income, and expenses. Next, you create a full-access user for your spouse. Afterwards, you create several read-only users for key family members and the executor of your estate. You also associate your health care agent with your medical directives. If you die unexpectedly or become incapacitated these users can access your information.

Scenario: After your family attorney registers with Estate⁺⁺ as an advisor you associate the attorney's firm with your Legal Dropbox. While creating your estate plan, your attorney uploads the initial draft of your estate plan to your Legal Dropbox folder. You download this document, review it, make changes, and upload it back to the Legal Dropbox. Afterwards, your attorney downloads your changes for review, etc... This exchange continues until the document is complete. Once complete, you move the document to a private folder.

Scenario: You have a new family attorney who needs to review your living will but you do not want to give this attorney access to your Legal Dropbox. You create a direct link to the document and email it to the attorney. The attorney is able to download the document for review. The direct link expires in 1 hour to prevent any unauthorized access to the document.

Scenario: You and your wife are travelling abroad and a thief snatches your wife's purse containing your passports, checkbook, and credit cards. You return to your hotel and use the hotel's computer to access Estate⁺⁺. You are able to quickly cancel your credit cards, freeze your checking account, and get a replacement passport before any identity theft occurs.

Scenario: Your software creates tax returns. When complete, the user would like to store a copy of the latest tax return in Estate⁺⁺. The user navigates to the file menu and clicks on the "Save to Estate⁺⁺" menu item. Sometime later the user has misplaced the tax return and cannot find it anywhere. The user simply logs into the Estate⁺⁺ website, navigates to the "Tax Returns/2009" folder, and downloads the document.

Before using the APIs

When an application stores access identifiers on a local computer, it's highly recommended that the private key be encrypted. The private key is a secret, and should be known only by the user and Estate⁺⁺. A user should never e-mail the private key to anyone. It is important to keep the private key confidential in order to protect the user's account. An application should warn the user to generate a new private key if its secrecy has been compromised in any way.

In order to use the Estate⁺⁺ APIs your application will need to prompt the user for the access identifiers, possibly as part of a "Customize..." dialog.

UploadByFolderName API

The UploadByFolderName API uploads files to vendor specific folders. If the folder(s) do not exist, they are created by the API. These folders are effectively private to a single vendor. If a file already exists in the folder with the same name, it will be deleted and replaced by the upload process. This is a REST-ful API so input parameters are provided as request headers and the uploaded file is attached to the request body as a multi-part mime attachment.

1. **Create a digital signature of the file being uploaded.** A digital signature is needed to ensure that the file has not been modified in transit or during subsequent downloads. The digital signature is an MD5 hash of input file. The hash is converted to a Base64 string encoded as US-ASCII.
2. **Create request signature.** The request signature is needed to ensure that the request has not been modified in transit. The request signature is an MD5 hash of all the request header data values concatenated as a string in the following order:
 Access Identifier Private Key + Access Identifier Public Key + Digital Signature
 + Full local file system path of the file being uploaded + Upload Folder Path + Vendor API Key
 The hash is converted to a Base64 string encoded as US-ASCII.
3. **Build the request headers** according to this table.

Request Header	Value
epp-x-pkey	Access identifier public key
epp-x-fn	Full local file system (LFS) path of the file to be uploaded. Examples: (Windows) C:\My Documents\EstatePlans\LivingWill.doc (UNIX) /usr/does/work/index.html (Macintosh) filemac:/MacintoshHD/Users/Documents/test.xls <i>Avoid using relative paths or UNC paths as they may yield unexpected results.</i>
epp-x-fd	A series of folders delimited by either a pipe character, a forward slash character, or a back slash character. Examples: /Estate Plans/Our Family/Wills/ Estate Plans Our Family Wills Estate Plans Our Family Wills Estate Plans/Our Family/Wills \Estate Plans\Our Family\Wills\ Estate Plans\Our Family\Wills Vendors should use folder names unique to their own product lines or a HTTP Status Code 409 Conflict may result. For example: /My Tax Filer/Tax Year 2010>Returns <i>Avoid using relative paths, local file system paths, or UNC paths as these may yield unexpected results.</i>
epp-x-vkey	Vendor API Key
epp-x-sig	Request signature

4. **Upload the file.** The file to be uploaded must attached to the request as a multi-part mime attachment. Use a language dependent method to send the REST-ful request to the following URI:
<https://www.estateplusplus-0020.net/uvault/o/UploadByFolderName.aspx>
5. **Process any errors.** Handle the HTTP status code according to the API error code table. API errors are returned within an XML document located in the Response body. For example:

```
<Error><Code>InternalServerError</Code>
<Description>Unable to delete from database</Description></Error>
```

UploadByFileSystemPath API

The UploadByFileSystemPath API uploads files to a folder structure similar to the one used by the file being uploaded. This API tries to mimic the local file system by allowing multiple vendors to access the same folder. For example:

- This file **C:\My Documents\EstatePlans\LivingWill.doc** is stored in the following folder structure **/My Documents/EstatePlans**
- This file **/usr/does/work/index.html** is stored in the following folder structure **/usr/does/work**
- This file **filemac:/MacintoshHD/Users/Documents/test.xls** is stored in the following folder structure **/MacintoshHD/Users/Documents**

If the folder(s) do not exist, they are created by the API. If a file already exists in the folder with the same name, it will be deleted and replaced by the upload process. This is a REST-ful API so input parameters are provided as request headers and the uploaded file is attached to the request body as a multi-part mime attachment.

1. **Create a digital signature of the file being uploaded.** A digital signature is needed to ensure that the file has not been modified in transit or during subsequent downloads. The digital signature is an MD5 hash of input file. The hash is converted to a Base64 string encoded as US-ASCII.
2. **Create request signature.** The request signature is needed to ensure that the request has not been modified in transit. The request signature is an MD5 hash of all the request header data values concatenated as a string in the following order:
Access Identifier Private Key + Access Identifier Public Key + Digital Signature
+ Full local file system path of the file being uploaded + Vendor API Key
The hash is converted to a Base64 string encoded as US-ASCII.
3. **Build the request headers** according to this table.

Request Header	Value
epp-x-pkey	Access identifier public key
epp-x-fn	Full local file system (LFS) path of the file to be uploaded. Examples: (Windows) C:\My Documents\EstatePlans\LivingWill.doc (UNIX) /usr/does/work/index.html (Macintosh) filemac:/MacintoshHD/Users/Documents/test.xls <i>Avoid using relative paths or UNC paths as they may yield unexpected results.</i>
epp-x-vkey	Vendor API Key
epp-x-sig	Request signature

4. **Upload the file.** The file to be uploaded must attached to the request as a multi-part mime attachment. Use a language dependent method to send the REST-ful request to the following URI:
<https://www.estateplusplus-0020.net/uvault/o/UploadByFileSystemPath.aspx>
5. **Process any errors.** Handle the HTTP status code according to the API error code table. API errors are returned within an XML document located in the Response body. For example:
<Error><Code>InternalServerError</Code>
<Description>Unable to delete from database</Description></Error>

API Error Code Table

The following status codes can be returned by either API. Errors are returned within an XML document located in the Response body. For example:

```
<Error>
<Code>InternalError</Code>
<Description>Unable to delete from database</Description>
</Error>
```

HTTP Status Code	Error Code	Description
400 Bad Request	ExpiredToken	Amazon subscription reactivation needed. Contact technical support. (support@estateplusplus.com)
400 Bad Request	EntityTooLarge	Upload file size exceeds available space for account. Upgrade account for additional upload space.
403 Forbidden	RequestSignatureDoesNotMatch	Request modified after being sent.
403 Forbidden	InvalidToken	Public key is invalid.
403 Forbidden	InvalidToken	Vendor API key is invalid.
409 Conflict	TopFolderNameAlreadyExists	The requested top folder name is not available because it is already in use. Please select another top folder name. (UploadByFolderName API only)
500 Internal Server Error	InternalError	Upload failed due to internal error. Description varies depending on error. Contact technical support. (support@estateplusplus.com)