Parallelspace Corporation Technical Note TN0016

SoftPass 0.5 Implementation Guide

SoftPass Version 0.5.6

Document Version 1.0

Michael Herman
Parallelspace Corporation
mwherman@parallelspace.com

Abstract

This document provides an overview of the technical deliverables for SoftPass 0.5 – version 0.5 of Parallelspace's peer activated software services solution.

CONTENTS

SOFTPASS 0.5 ARCHITECURE AND DELIVERABLES

The business and technical goals of SoftPass 0.5 are to enable Software Developers with a rich set of peer activated software services that can be integrated into their products and deployed as quickly as possible.

The technical deliverables consist of 3 components:

1. SoftPass 0.5 client DLL
2. SoftPass Software Activation client tool
3. SoftPass Issuer back-end services

The SoftPass 0.5 client DLL exports methods that a developer can call from their initialization code (or anywhere in their code) to determine if the Software Product has been activated for use by the current user and if so, which edition(s) of the software product the end user is entitled to use. The developer of the software product initializes the appropriate internal or global data structures as required to control how a product behaves for the user. Of course, this should be based on the edition of the product the user is entitled to use. SoftPass 0.5 provides support for the following editions of a software product. Additional editions can be added in SoftPass 1.0. It is entirely up to the Software Developer to determine which editions and edition names make sense for their products and their customers.

Preview	Bronze
Standard	Silver
Professional	Gold
Advanced	Platinum
Server	Titanium

With SoftPass 0.5, Customers purchase Software Solutions directly from the Software Developer or a Software Developer distributor or dealer (i.e. no eCommerce services are available with SoftPass 0.5 but they will be available with SoftPass 1.0). As part of the purchase transaction, the Customer agrees to the traditional End User License Agreement (EULA) terms and conditions and downloads and injects the Software Solution (and component products) in whichever order makes sense for the Software Developer, their Software Solutions and Products as well as their Customers. The EULA can be presented immediately as part of the purchase transaction, as a pre-requisite before downloading the software product(s) or when asked to after the software product has been downloaded, installed and run for the first time.

A Software Solution can consist of a single product or possibly multiple products from multiple Software Developers. In SoftPass 0.5, a Software Solution can include one, two or three Software Products.

The SoftPass client DLL will search all of the client machine's SoftPasses for the first SoftPass that has a matching GrooveIdentity URN, product name, product version and if specified, a particular product

edition. If the product edition is not part of the query (i.e., the product edition parameter is set to zero), the activation method call will return a union of all of the SoftPass editions that match the requested GrooveIdentity URN, product name, and product version.

By default, the freshly installed software product(s) should have a default set of partial (or complete) functionality enabled that allows both a solitary user as well as a team of users to experience and evaluate the Software Solution. The usual expectation is that the default set of functionality will be the same as for the Preview edition of the product.

At this point, the software product(s) for the Software Solution purchased by the Customer have been downloaded and installed but the Customer or End User has not activated the software; hence, only the Preview level or subset of the functionality is available. The End User now needs to activate the solution to enable the set of functionality that corresponds to the edition of the product they have purchased. The End User uses the SoftPass Activation tool to do this.

The SoftPass Activation tool is a free Groove Workspace tool that uses a wizard to lead End Users through the steps of the software activation process. In SoftPass 0.5, an End User can request activation for a single user: themselves. In SoftPass 1.0, an End User will be able to request activation for any list of Groove users.

Why a Groove tool? The SoftPass Activation tool is implemented as a Groove tool for the following reasons:

- A SoftPass is issued to a particular GrooveIdentity URN and the only practical, user friendly way to provide this information to the SoftPass Issuer back-end is through a Groove tool.
- To enable activation of more than one user in SoftPass 1.0, the user needs to be able to select a list of users in much the same way they select the list of users to invite into a shared space or to send a Groove message to.
- Future versions of SoftPass will also enable activation from directly with a Software Developer's Software Product.

A SoftPass activation request includes the End User's GrooveIdentity, name, affiliation, address, email address, Software Solution name, version and edition information as well as the start date and duration time period to be used to create the End User's SoftPass.

The SoftPass Issuer back-end services component is an Internet-hosted XML web services server that receives the activation requests from the SoftPass Activation tool and forwards the request to the Software Developer or Software Publisher for authorization to create and issue a SoftPass for the specific Groove user, Software Solution, version and edition. Upon receiving authorization from the Software Developer or Software Publisher responsible for the Software Solution, the SoftPass Issuer will send a notification to the End User advising them that a SoftPass has been issued to them for the particular Software Solution as

well as information about how to download and install the SoftPass. This completes the SoftPass software activation process.

WHAT IS SOFTPASS?

Parallelspace SoftPass is a peer activated software services solution for P2P, decentralized software solutions. The business and technical goals of SoftPass 0.5 are to enable Software Developers with a rich set of peer activated software services that can be integrated into their products and deployed as quickly as possible.

A decentralized software solutions business ecosystem needs to include:

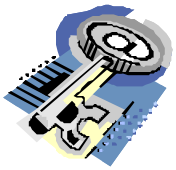
1. Software Developers who conceive, develop, own and service their Software Solutions
2. Software Publishers who market, sell and provide first level support for the Software Solutions
3. Software eCommerce Partner(s) who provide the eCommerce infrastructure for Customer wishing to purchase a Software Solution from a Software Publisher
4. SoftPass Issuer(s) who operate the Software Activation Clearinghouse that provides software activation services for Software Developers, Software Publishers and their Customers.

A software developer or publisher might choose to provide all of the above services or alternatively choose to delegate one or more of these services to one or more partner companies.

Software Developer/OEM

Software Publisher

Software eCommerce Partner



Software Activation Partner

Customer

Goals of SoftPass

The business and technical goals of SoftPass 0.5 are to enable Software Developers with a rich set of peer activated software services that can be integrated into their products and deployed as quickly as possible.

TECHNICAL ARCHITECTURE

The technical deliverables consist of 2 components:

1. SoftPass 0.5 client DLL
2. SoftPass Groove Script Library (.gsl)

The SoftPass 0.5 client GSL exports methods that a developer can call from their initialization code (or anywhere in their code) to determine if the Software Product has been activated for use by the current user and if so, which edition(s) of the software product the end user is entitled to use. The developer of the software product initializes the appropriate internal or global data structures as required to control how a product behaves for the user. Of course, this should be based on the edition of the product the user is entitled to use. SoftPass 0.5 provides support for the following editions of a software product. Additional editions can be added in SoftPass 1.0. It is entirely up to the Software Developer to determine which editions and edition names make sense for their products and their customers.

Preview	Bronze
Standard	Silver
Professional	Gold
Advanced	Platinum
Server	Titanium

The SoftPass client DLL will search all of the client machine's SoftPasses for the first SoftPass that has a matching GrooveIdentity URN, product name, product version and if specified, a particular product edition. If the product edition is not part of the query (i.e., the product edition parameter is set to zero), the activation method call will return a union of all of the SoftPass editions that match the requested GrooveIdentity URN, product name, and product version.

USING SOFTPASS IN YOUR GROOVE WORKSPACE TOOL

SoftPass is a natural addition to any Groove Workspace Tool considering the P2P nature of Groove. It facilitates the easy distribution of your application and ensures that your application is always appropriately licensed.

Best Practices

To ensure that SoftPass has not be been tampered with we highly recommend that you follow the steps below. After initializing the SoftPass DLL you should check that the DLL has been signed and can be trusted using SoftPassVerifyDLLFromProgID(ProgID). Once that is successful you should ensure that the Hash value of the DLL is consistent with the SoftPass DLL shipped so you should call SoftPassVerifyHash(ProgID).

Once you have completed these steps and everything has returned successfully only then can you trust the DLL and the values it returns. If any of the above functions fail the SoftPass DLL has been tampered with and should not be trusted.

Sample Code

```
// Initialize the SoftPass
if( SoftPassInitialize() == gkfSoftPass_Success )
{
  // Check if the DLL is signed
  if( SoftPassVerifyDLLFromProgID( "PSNSoftPass.SoftPass" ) ==
gkfSoftPass_Success )
  {
    // Verify the DLL's Hash
    if( SoftPassVerifyHash( "PSNSoftPass.SoftPass" ) ==
gkfSoftPass_Success )
    {
      // Check the users activation level
      fActivation = SoftPassCheckActivation( lAuthType, sUserId,
                                        sPassword, sProduct,
                                        sVersion, lfEditions );
      if ( (fActivation == gkfSoftPass_ActivationDenied) ||
                        (fActivation == gkfSoftPass_BadDLL) ||
           (fActivation == gkfSoftPass_ActivationExpired) )
      {
        fActivation |= gkfSoftPass_ActivationPre;
        lReturnCode = 1;
      }
    }
    else
    {
      fActivation = gkfSoftPass_ActivationPre | gkfSoftPass_BadDLL;
      lReturnCode = 0;
    }
  }
}
```

SOFTPASS FUNCTION DOCUMENTATION

function SoftPassInitialize()

SoftPassInitlialize must be called prior to any other call to SoftPass. SoftPassInitilize first creates the SoftPass DLL reference. This function also iniliazes internal sturtures within the SoftPass DLL.

Parameters

None.

Returns

gkfSoftPass_Success – On success

gkfSoftPass_Failure – On failure

Notes:

gkfSoftPass_Failure usually occurs if the SoftPass DLL has not been registered properly or has been corrupted.

---

function SoftPassTerminate()

SoftPassTerminiate ensures that all the structures used in the SoftPass DLL and are deleted and all memory allocated is freed.

Parameters

None.

Returns

gkfSoftPass_Success – On success

gkfSoftPass_Failure – On failure

Notes:

None.

---

function SoftPassCheckActivation(lAuthType, sUserId, sPassword, sProduct, sVersion, lfEditions)

SoftPassCheckActivation enumerates all the SoftPass this user has then searches them for the criteria specified in the call.

Parameters

lAuthType [in]

Type – Long

Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]
Type – String
Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sPassword [in]
Type – String
Description – This is the user's password, required for gkfSoftPass_SoftPassAuthType_Email, gkfSoftPass_SoftPassAuthType_Passport. Reserved for future user do not use, must be empty string.

sProduct [in]
Type – String
Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]
Type – String
Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lfEditions [in]
Type – Long
Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. Optionally this parameter can be 0(zero) which will have this function return all valid editions available to this user.

Returns

If the value for *lfEditions* is 0 then the following values will be bitwise or'ed together.

Value	Meaning
gkfSoftPass_ActivationDenied	This will be returned if a SoftPass can not be found with the parameters specified in the function call.
gkfSoftPass_ActivationPre	This will be returned if the preview edition is authorized for the

	parameters specified in the function call.
gkfSoftPass_ActivationStd	This will be returned if the standard edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationPro	This will be returned if the professional edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationAdvanced	This will be returned if the advanced edition is authorized for the parameters specified in the function call. This will be returned if the preview edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationServer	This will be returned if the server edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationBronze	This will be returned if the bronze edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationSilver	This will be returned if the silver edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationGold	This will be returned if the gold edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationPlatinum	This will be returned if the platinum edition is authorized for the parameters specified in

	the function call.
gkfSoftPass_ActivationTitanuium	This will be returned if the titanium edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationExpired	This will be returned if the user has SoftPass but it has expired for the parameters specified in the function call.
gkfSoftPass_BadDLL	If the DLL is corrupt.
gkfSoftPass_ActivationOverdrive	Reserved for future use
gkfSoftPass_ActivationFutureVersion	Reserved for future use

Notes:

None.

---

function SoftPassGetProductEditions(lAuthType, sUserId, sProduct, sVersion)

SoftPassGetProductEditions returns the editions this person is authorized for.

Parameters

lAuthType [in]

Type – Long

Description – Flag value indicating the authentication type

Value	**Meaning**
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]

Type – String

Description – The users Groove Identity URL. The user follows the format “grooveIdentity://....”.

sProduct [in]

Type – String

Description – This parameter is required and should be your products name as specified in the SoftPass. The

comparison is case sensitive.

sVersion [in]
Type – String
Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

Returns

If more than one edition is authorized for this user then the return value will be bitwise or'ed together.

Value	Meaning
gkfSoftPass_ActivationDenied	This will be returned if a SoftPass can not be found with the parameters specified in the function call.
gkfSoftPass_ActivationPre	This will be returned if the preview edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationStd	This will be returned if the standard edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationPro	This will be returned if the professional edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationAdvanced	This will be returned if the advanced edition is authorized for the parameters specified in the function call. This will be returned if the preview edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationServer	This will be returned if the server edition is authorized for the parameters specified in

	the function call.
gkfSoftPass_ActivationBronze	This will be returned if the bronze edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationSilver	This will be returned if the silver edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationGold	This will be returned if the gold edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationPlatinum	This will be returned if the platinum edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationTitanuium	This will be returned if the titanium edition is authorized for the parameters specified in the function call.
gkfSoftPass_ActivationExpired	This will be returned if the user has SoftPass but it has expired for the parameters specified in the function call.
gkfSoftPass_BadDLL	If the DLL is corrupt.
gkfSoftPass_ActivationOverdrive	Reserved for future use
gkfSoftPass_ActivationFutureVersion	Reserved for future use

Notes:

None.

---

function SoftPassGetProductSolutionCount(lAuthType, sUserId, sProduct, sVersion, lEdition)

SoftPassGetProductSolutionCount returns the number of the solutions that this product and version belong to. This function is used so that a

developer can enumerate through solutions.

Parameters

lAuthType [in]

Type – Long

Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]

Type – String

Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sProduct [in]

Type – String

Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]

Type – String

Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]

Type – Long

Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. This parameter must be set.

Returns

A zero based index of all the solutions that the specified product is associated with.

Notes:

None.

function SoftPassGetProductSolutionName(lAuthType, sUserId, sProduct, sVersion, lEdition, iSolutionIndex)

SoftPassGetProductSolutionName returns the name of the solution at iSolutionIndex.

Parameters

lAuthType [in]

Type – Long

Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]

Type – String

Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sProduct [in]

Type – String

Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]

Type – String

Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]

Type – Long

Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. This parameter must be set.

iSolutionIndex [in]

Type – Integer

Description – This parameter

Returns

The solution name for the given index value.

Notes:

None.

---

function SoftPassGetProductSolutionVersion(lAuthType, sUserId, sProduct, sVersion, lEdition, iSolutionIndex)

SoftPassGetProductSolutionVersion returns the solution version at iSolutionIndex.

Parameters

lAuthType [in]
Type – Long
Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]
Type – String
Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sProduct [in]
Type – String
Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]
Type – String
Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]
Type – Long
Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. This parameter must be set.

iSolutionIndex [in]
Type – Integer
Description – This parameter

Returns

The solution's version for the given index value.

Notes:

None.

---

function SoftPassGetProductSolutionOEMOrgFriendlyName(lAuthType, sUserId, sProduct, sVersion, lEdition, iSolutionIndex)

SoftPassGetProductSolutionOEMOrgFriendlyName returns the OEM Friendly Name of the solution at iSolutionIndex.

Parameters

lAuthType [in]

Type – Long

Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]

Type – String

Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sProduct [in]

Type – String

Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]

Type – String

Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]

Type – Long

Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd.

This parameter must be set.

iSolutionIndex [in]
Type – Integer
Description – This parameter

Returns

The solution's OEM friendly name for the given index value.

Notes:

None.

---

function SoftPassGetProductSolutionSKU(lAuthType, sUserId, sProduct, sVersion, lEdition, iSolutionIndex)

SoftPassGetProductSolutionSKU returns the SKU of the solution at iSolutionIndex.

Parameters

lAuthType [in]
Type – Long
Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]
Type – String
Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sProduct [in]
Type – String
Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]
Type – String
Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]
Type – Long
Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. This parameter must be set.

iSolutionIndex [in]
Type – Integer
Description – This parameter

Returns

The solution's SKU number for the given index value.

Notes:

None.

---

function SoftPassGetProductSolutionURN(lAuthType, sUserId, sProduct, sVersion, lEdition, iSolutionIndex)

SoftPassGetProductSolutionURN returns the URN of the solution at iSolutionIndex

Parameters

lAuthType [in]
Type – Long
Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]
Type – String
Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sProduct [in]
Type – String
Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]
Type – String
Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]
Type – Long
Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. This parameter must be set.

iSolutionIndex [in]
Type – Integer
Description – This parameter

Returns

The solution's URN for the given index value.

Notes:

None.

---

function SoftPassGetProductSolutionUserSector(lAuthType, sUserId, sProduct, sVersion, lEdition, iSolutionIndex)

SoftPassGetProductSolutionURN returns the user sector of the solution at iSolutionIndex.

Parameters

lAuthType [in]
Type – Long
Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]
Type – String
Description – The users Groove Identity URL. The user follows the format "grooveIdentity://....".

sProduct [in]

Type – String
Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]
Type – String
Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]
Type – Long
Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. This parameter must be set.

iSolutionIndex [in]
Type – Integer
Description – This parameter

Returns

The solution's user sector for the given index value.

Notes:

None.

---

function SoftPassGetProductSolutionEdition(lAuthType, sUserId, sProduct, sVersion, lEdition, iSolutionIndex)

SoftPassGetProductSolutionURN returns the edition of the solution at iSolutionIndex.

Parameters

lAuthType [in]
Type – Long
Description – Flag value indicating the authentication type

Value	Meaning
gkfSoftPass_SoftPassAuthType_Groove	
gkfSoftPass_SoftPassAuthType_Email	Reserved for future use do not use
gkfSoftPass_SoftPassAuthType_Passport	Reserved for future use do not use

sUserId [in]

Type – String
Description – The users Groove Identity URL. The user follows the format “grooveIdentity://....”.

sProduct [in]
Type – String
Description – This parameter is required and should be your products name as specified in the SoftPass. The comparison is case sensitive.

sVersion [in]
Type – String
Description – This parameter is required and should be your products version as specified in the SoftPass. The comparison is case sensitive.

lEdition [in]
Type – Long
Description – This parameter can be the specific edition you are looking for example gkfSoftPass_ActivationStd. This parameter must be set.

iSolutionIndex [in]
Type – Integer
Description – This parameter

Returns

The solution’s edition for the given index value.

Notes:

None.

---

function SoftPassGetDLLVersionFromPath(sDLLPath)

SoftPassGetDLLVersionFromProgID returns the file version of the file given a path (sDLLPath).

Parameters

sDLLPath [in]
Type – String
Description – The path to the file.

Returns

The files version information.

Notes:

None.

function SoftPassGetDLLVersionFromProgID(sDLLProgID)

SoftPassGetDLLVersionFromProgID returns the file version of the file given a ProgID.

Parameters

sDLLProgID [in]

Type – String

Description – The ProgID of the DLL.

Returns

The files version information.

Notes:

None.

---

function SoftPassVerifyDLLFromPath(sDLLPath)

SoftPassGetDLLVersionFromPath verifies whether a file is signed given a file path.

Parameters

sDLLPath [in]

Type – String

Description – The path to the file.

Returns

gkfSoftPass_Success – On success

gkfSoftPass_Failure – On failure

Notes:

None.

---

function SoftPassVerifyDLLFromProgID(sDLLProgID)

SoftPassVerifyDLLFromProgID verifies whether a file is signed given a ProgID.

Parameters

sDLLProgID [in]

Type – String

Description – The ProgID of the DLL.

Returns

gkfSoftPass_Success – On success

gkfSoftPass_Failure – On failure

Notes:

None

function SoftPassVerifyHash(sDLLProgID)

SoftPassVerifyHash returns the files hash value given a ProgID.

Parameters

sDLLProgID [in]

Type – String

Description – The ProgID of the DLL.

Returns

gkfSoftPass_Success – On success

gkfSoftPass_Failure – On failure

Notes:

None