|  |
| --- |
| Cargo Community Network Pte Ltd |
| Collaboration Platform (CP) API Guide |
|  |
|  |
| **Koh Chit Hwee** |
| **Sep 2014** |

Contents

[1.Version and Changes 1](#_Toc411005969)

[2.Introduction 2](#_Toc411005970)

[3.CP Interface Information 4](#_Toc411005971)

[3.1 ProductAPIService Information 8](#_Toc411005972)

[3.1.1 SecureProductAPIService Information 10](#_Toc411005973)

[3.2 Classic aspService Information 11](#_Toc411005974)

[4.Serivces URL Information 13](#_Toc411005975)

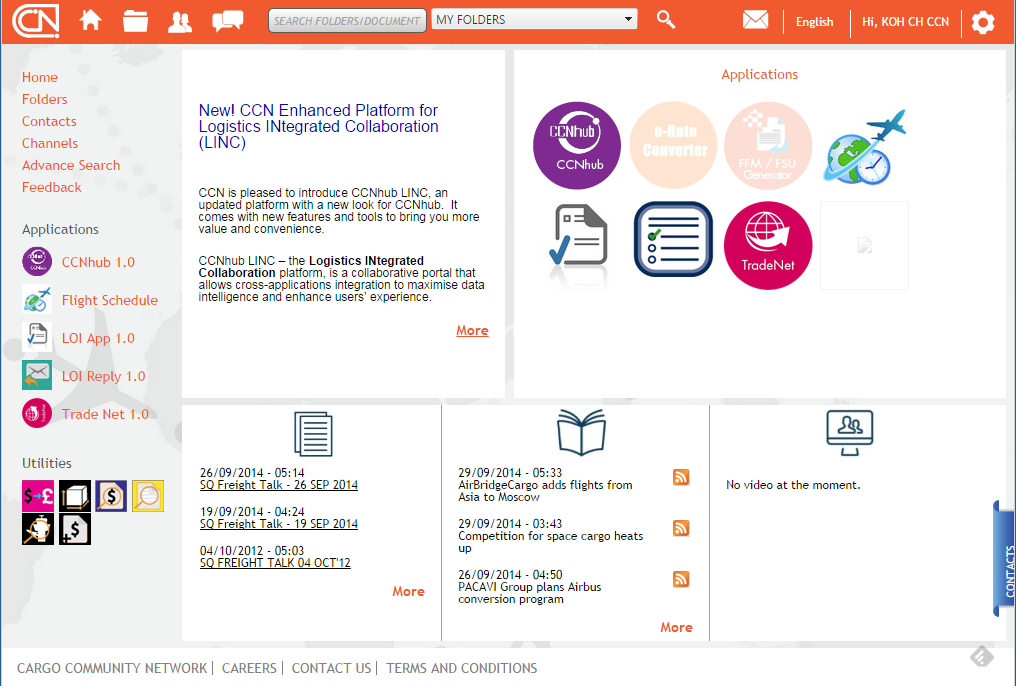
[5.CP Application Information 14](#_Toc411005976)

# Version and Changes

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Changes** | **Documented by** |
| 1.00 | 08 Apr 2014 | Initial Copy. | Koh Chit Hwee |
| 1.10 | 30 Sep 2014 | Update for New API methods.  Add section 3.1 | Koh Chit Hwee |
| 1.11 | 7 Oct 2014 | Update the Product Table for ProductNewConfigURL and ProductEditConfigURL. | Koh Chit Hwee |
| 1.2 | 28 Nov 2014 | Update section 3 and 3.1 for new API methods   * SaveProductConfigSetting * CreateUserAccount * SaveUserAccount * DeleteUserAccount   Update section 3.1 for change in GetCompanyPima.  New fields for DTOUserAccount and DTOUserProduct. | Koh Chit Hwee |
| 1.3 | 12 Jan 2015 | Update section 2 for summary diagram.  Update section 3 and 3.1 for new IHierarchy interface and new method GetUserAccountsByEmail.  Add section 3.2 for Classic aspService.aspx methods.  Update section 4 for HierarchyService URL. | Koh Chit Hwee |
| 1.4 | 6 Feb 2015 | Update section 3.1 for the UpdateStation in ProductAPIService.  Update existing section and add section 3.1.1 for SecureProductAPIService.  Update section 4 for SecureProductAPIService URL. | Koh Chit Hwee |

# Introduction

This document is to provide information for interface API needed to integrate .NET application into CP Landing page. The CP Landing page is shown below where there is different application displaying on the page for user to access their application. In order for the application to work in the CP Landing page, the application needs to follow the following section to interface with CP Landing page. The application for the CP Landing page is display in the main panel and the left panel as shown below. The utility applications are also display on the left panel below the Application list under Utilities.



For application that need integrate to CP Landing page, there are interface and methods for it’s to consume. This is provided in the following section as below:

Section 3.0 – Provide the interface to implement by the application for CP to call.

Section 3.1 – Provide the methods in the ProductAPIService.

Section 3.1.1 – Provide the detail for the SecureProductAPIService.

Section 3.2 – Provide the methods in the aspService.aspx.

Section 4.0 – Provide the URL information for the service / servers

The summary diagram on next page shows the interface and service for the application to integrate with CP that is provided in this document.

**Summary Diagram for integrating Application in CP**

Call provided methods

**Application to integrate in CP**

Implement interface needed and call ProductAPIService

**CP Landing page**

When click on Application icon to launch, it call IProduct.**ValidateAccess**

**CP User Management page**

When assign Application to the User, it call IProduct.**Activate**.

When unassign Application to User, it call IProduct.**Terminate**

Optional Implement IHierarchy interface

Refer to 3.0

**ProductAPIService**

WCF Service

Refer to 3.1

**CP PortalAdmin / CP Product Management page**

When assign Application to Station, it call IHierarchy.ActivateStationProduct

This is to expose methods for .NET application to consume that need to integrate in CP.

This is to expose methods for classic ASP application to consume that need to integrate in CP.

**aspService.aspx page**

*POST to this page*

Refer to 3.2

Implement IProduct interface to perform own logic when called.

Refer to 3.0

R

**SecureProductAPIService**

WCF Service

Refer to 3.1.1

# CP Interface Information

The interface API needed to integrate .NET application into CP Landing page is to create its application AppProductService that implement the ProductService.IProduct interface as shown below. Also application that needs Company Hierarchy to implement the HierarchyService.IHierarchy interface as shown below. Next the application also needs to call the CP WCF Service which is ProductAPIService to create log for tracking the user accessing the application and to check the CP Session is active and to keep CP Session alive. Last the application need to provide configuration for the Product table to display the application in the CP Landing page.

**Application Product Service**

**ProductService**

Refer to Services URL information

**The ProductService.IProduct defines the following methods for other application to implement:**

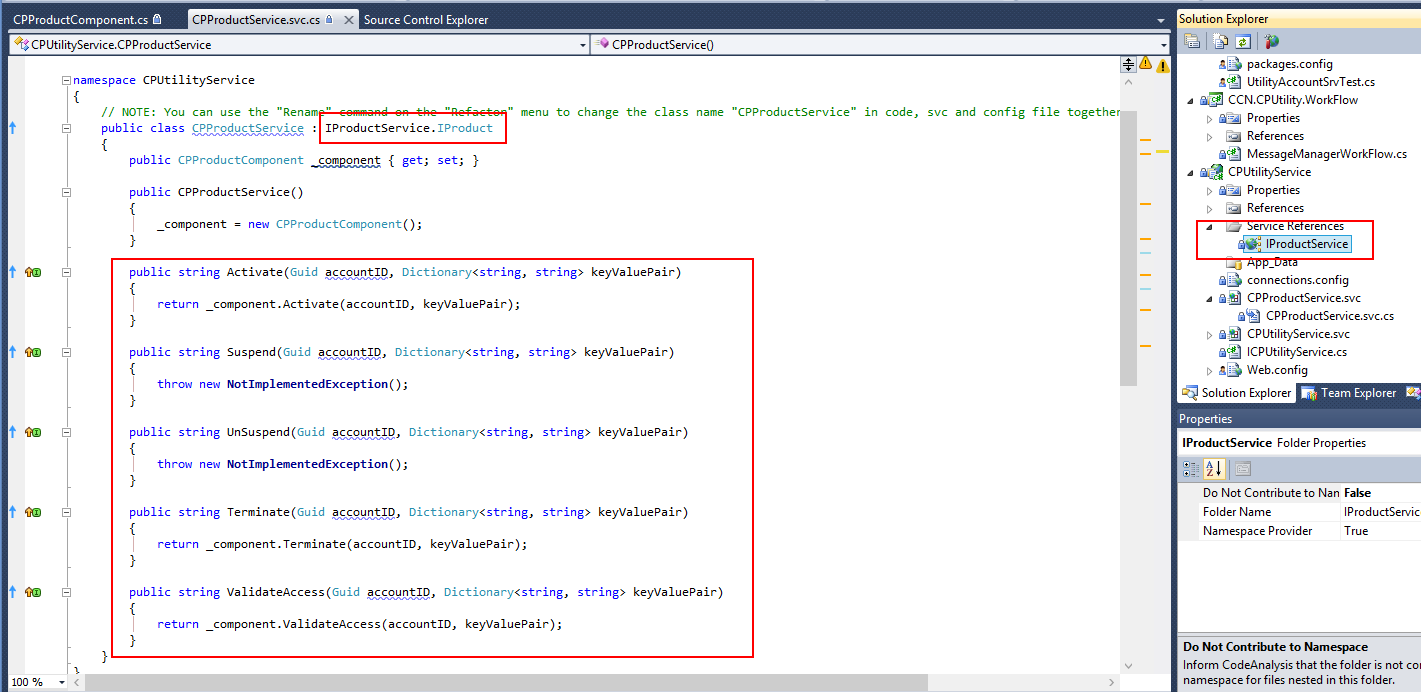
* Activate
* Suspend
* UnSuspend
* Terminate
* ValidateAccess

IProduct

*The application that needs to interface to the CP Landing page must create a WCF service that implement the IProduct interface from the ProductService.*

Implement ProductService.IProduct

Below show an example of an application Service that adds ProductService interface to implement the required methods.



The implement methods will be called from the CP Landing for the following case:

When new user account in CP created, if it is a default application, it will call the **Activate** method. This is to let the application know the new AccountID for the account created. Otherwise, the activate method will only be called when user subscribe to it. When the Activate method is called, the keyvalue pair pass to the method consists of the below keys:

{"**ProductID**",item.ProductID},

{"**GlobalCompanyID**",item.GlobalCompanyID.ToString()},

{"**City**",item.City}

When account in CP is suspend, unsuspend or terminate it will call the respective method to let the application know the status of the user account. It is the same when user unsubscribe in the app store. So there are also 3 image files for the application status that is Activated, Suspended and Terminated to display on CP Landing page.

When the user click the icon on the CP Landing page, the **ValidateAccess** will call the application to validate the account is allowed to access or return the error string to denied access to the application. The validate Access method need to call **ProductAPIService.ProductStartLog** (below) to create the session for the application and its own validation method before returning success.

For application that need to manage Company Hierarchy for its own purpose will need to implement the HierarchyService.IHierarchy as shown below, this is optional for application that do not need company hierarchy.

The interface currently use in CP is the **ActivateStationProduct**. This is called from CP when Helpdesk or HQ Admin assigned the application to the Station in CP. The ActivateStationProduct method is called to inform the application that it is had assigned to a Station in the Company, so that application can manage the Hierarchy / Company info within the method. So for the application to get the Company / Hierarchy info from CP, it can use the methods provided in the ProductAPIService as shown in next section.

**The HierarchyService.IHierarchy defines the following methods for other application to implement:**

* ActivateStationProduct
* TerminateStationProduct

**Application Product Service**

**HierarchyService**

Refer to Services URL information

IHierarchy

*The application that needs to interface to the CP for Company Hierarchy in their application will create a WCF service that implement the IHierarchy interface from the HierarchyService.*

Implement HierarchyService.IHierarchy

**Note: For the TerminateStationProduct it is for un-assigning the application in CP for the station. But this is not called in CP as this feature not in CP yet.**

CP also provides another WCF Service for other application to call for integration, the ProductAPIService as shown below provide methods to support the following function as in section 3.1:

**ProductAPIService**

Refer to Services URL information

**The ProductAPIService defines the following methods for other application to call:**

* ProductStartLog
* ProductEndLog
* IsSessionActive
* GetCompanyHierarchy
* GetCompany
* GetStation
* GetUserAccounts
* GetUserAccountsByEmail
* GetUserAccount
* GetCompanyRoles
* GetStationRoles
* GetCompanyPima
* GetPima
* SaveProductConfigSetting
* CreateUserAccount
* SaveUserAccount
* DeleteUserAccount
* UpdateStation

**Note: ProductAPIService is for internal app that can access the CCN internal URL**

**For Application that needs to use the ProductAPIService using the public URL should use the SecureProductAPIService**

The SecureProductAPIService provide the same methods available in the ProductAPIService, but to call the methods in the SecureProductAPIService, the application need to send an encrypted ProductKey to access these methods.

For the SecureProductAPIService, please refer to the section 3.1.1 for detail how to call the method with the encrypted ProductKey.

For the application to display on the CP Landing page, it must be configured in the Product table in the UserAccount Database. Therefore, the application will need to provide these information to CP for configuration in the PortalAdmin.

|  |  |  |
| --- | --- | --- |
| **Table fields** | **Purpose** | **Remarks** |
| **ProductID** | **This is ID for the application** | **CP assign** |
| **ProductTitle** | **This is the application name to display on CP Landing page.** |  |
| **ProductDescription** | **The description of the application** |  |
| **ProductVersion** | **The version for the application** |  |
| **Publisher** | **The publisher for the application** |  |
| **ProductServiceURL** | **This is the WCF Service application URL that implements the ProductService.IProduct interface.** |  |
| **ProductImage** | **The image file name to display on the CP Landing page for the Main panel.**  **Image file is a png format size 90 X 90 pixel** | **Also the image files must be provided.** |
| **ProductIcon** | **The image file name to display on the CP Landing page for the Left panel.**  **Image file is a png format size 32 X 32 pixel** |  |
| **ProductURL** | **This is the URL of the application when user access from CP Landing page.** | **The URL can have querystring value for**  **[@SessionID]**  **[@AccountID]**  **[@ProductID] [@GlobalCompanyID]**  **[@City]** |
| **ProductNewConfigURL** | **This is the URL of the application for New Configuration page use by the CCN Helpdesk.** | **This is option if there not configuration for the application needed by Helpdesk.** |
| **ProductEditConfigURL** | **This is the URL of the application for Edit Configuration page use by the CCN Helpdesk.**  **(This maybe same as the ProductNewConfigURL if the page support edit configuration)** | **This is option if there not configuration for the application needed by Helpdesk.** |
| **ProductHierarchyConfigURL** | **This is the WCF Service application URL that implement the HierarchyService.IHierarchy interface.** | **This is option if the application does not implement the IHerarchy interface.** |
| **IsNewWindow** | **To indicate the ProductURL will open in another browser window.** |  |
| **IsDefault** | **To indicate the application is Activated when CP user account created.** |  |
| **HasStationConfig** | **To indicate whether the application need extra configuration setting when assign to new station.** |  |
| **IsAPIUpdateAllow** | **To indicate whether the SecureProductAPIService allow update methods to be called** |  |

# 3.1 ProductAPIService Information

This section is to provide a brief description for the WCF methods found in the ProductAPIService that the .NET application can called.

**ProductStartLog** is for the application to create a Tracking log for user to know when it starts access the application. When the application is launch from the CP Landing page, the application must call this method.

**ProductEndLog** is for the application to create the Tracking log for user to know when it logout from the application. When the application had logout / closed by the user, the application should call this method to let CP know to end the Product Session for the user.

**IsSessionActive** is for the application to update the user CP Session is still active, so it the CP Session still active when user using the application. It is also for the application to check whether the CP Session and Product Session had expired, so the application can end the user session. The method return an empty string if the CP session and Product Session are still active else an error string is return for the session.

**GetCompanyHierarchy** is for the application to get the Company Hierarchy for the GlobalCompanyID pass in. The List of DTO return the stations hierarchy for the company with the ControllingCity as the parent city for the station.

**GetCompany** is for the application to get the Company info for the GlobalCompanyID pass in. The DTO return the properties of the company info.

**GetStation** is for the application to get the Station info the the GlobalCompanyID and City pass in. The DTO return the properties of the station info.

**GetUserAccounts** is for the application to get the list of users for the GlobalStationID pass in. The DTO return the properties of the user account info.

**GetUserAccountsByEmail** is for the application to get the list of users for the email address pass in. The DTO return the properties of the user account info.

**GetUserAccount** is for the application to get the user account info for the AccountID pass in, the ProductID is pass as empty string if the user product info is not needed. The ProductID pass as “\*” will get all the user product info for the account, else it will get the user product info for the ProductID pass in.

**GetCompanyRoles** is for the application to get the Company Roles info detail for the GlobalCompanyID pass in. The list of DTO return the properties of the role detail for each role the company has.

**GetStationRoles** is for the application to get the Station Roles info detail for the GlobalCompanyID and City pass in. The list of DTO return the properties of the role detail for each role the station has.

**GetCompanyPima** is for the application to get the list of PIMA info for the GlobalCompanyID, City and ProductID pass in. The City and ProductID can be string.Empty if to get the list of PIMA for the GlobalCompanyID. To get list of PIMA for the station, then the values for the City have to pass in to return the list of DTO for the properties of PIMA info.

**GetPima** is for the application to get the list of PIMA info for the PIMA and Origin pass in. The Origin can be “\*” if to get all the PIMA for any origin City code, else it get for that origin City code pass in. The list of DTO returns the properties of the PIMA info like GlobalCompanyID and GlobalStationID.

**SaveProductConfigSetting** is for the application to notify CP for the setting to be saved when the application Product configuration is completed. So when Product configuration is saved, it will call to notify CP for the configuration done.

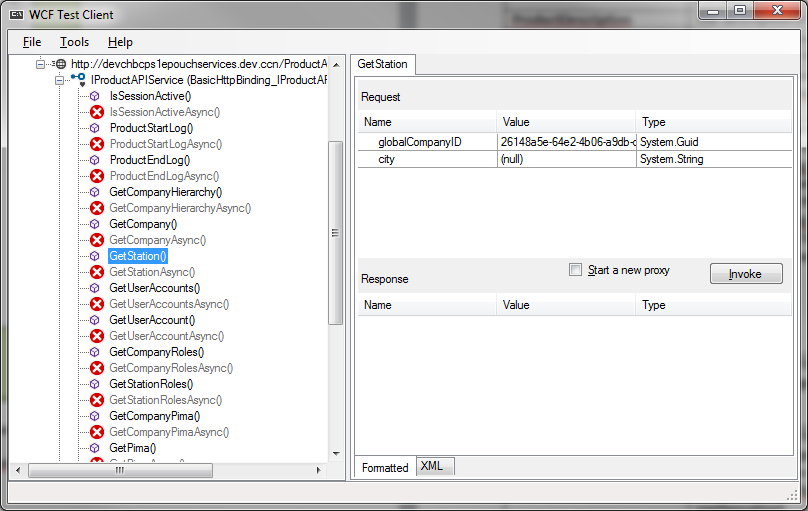
**CreateUserAccount** is for the application to create a new CP user account for the user to login into LINC.

**SaveUserAccount** is for the application to save the user account info when there changes.

**DeleteUserAccount** is for the application to delete the CP user account so that user will NOT login into LINC.

**UpdateStation** is for the application to update the station role profile info when there changes, if the station profile only has 1 Role, then it will also update the station detail. To update the station role profile and its station detail, the parameter for the RoleCodes role has to use the ENUM RoleCodes.OWN

To know more detail for the parameters for these methods in the ProductAPIService, you can use the WCF Test Client provided in the .NET Framework to see all the available methods as shown. So you can also use this Test Client to test the methods in the ProductAPIService.



Note for the SecureProductAPIService, you cannot use the WCF Test Client to test as it needs the additional encrypted ProductKey in the message header as mention in next section. But you can use the WCF Test Client to test the methods in the ProductAPIService as both services provide the same methods.

# 3.1.1 SecureProductAPIService Information

This section is to provide the detail for the SecureProductAPIService which has the same methods available in the ProductAPIService, but to call the methods in the SecureProductAPIService, the application need to send an encrypted **ProductKey** to access these methods as shown in the sample code below.

For the SecureProductAPIService, it is exposed to the public URL and to consume the methods, an encrypted **ProductKey** is needed to send in the messageHeader which the server will validate this **ProductKey** to allow the method to be executed. If the **ProductKey** is invalid, then the error below will be return.

The remote server returned an unexpected response: (400) Unknown Product Key.

To get this encrypted ProductKey, you need to format the productKey using the ProductID of the application and the timestamp of the current **UTC time** that is format as “*yyyyMMddHHmm*”. That is to append the UTC timestamp to the ProductID of the application as shown: string productKey = string.Format("{0}{1}", productId, timestamp);

Note the timestamp should be using UTC time NOT Local time if the wrong timestamp is pass to server validation, it will return the below error.

The remote server returned an unexpected response: (400) The request is out of date.

After this productKey is formatted, it will need to be encrypted using the Encrypt() method provided by the **ComponentEncryption.dll** which is provided by CCN. Also the encrypt method need a **passPhrase** which is provided by CCN, and you can save this in your database or app setting as shown string passPhrase = ConfigurationManager.AppSettings["PassKey"];

So with the encrypted **ProductKey**, then it can call the ProductAPIService methods by adding this encrypted productKey in the message header as shown in the sample code within the { } using (new OperationContextScope (client.InnerChannel))

For those **non Get** methods except for IsSessionActive, ProductStartLog and ProductEndLog, the application need to allow those methods to be executed, else the error below will be return.

The remote server returned an unexpected response: (400) Product is not allowed to update.

So CCN will configure whether the application can execute these methods in the API.

**Sample code**

string timestamp = string.IsNullOrEmpty(DateTime.UtcNow.ToString("yyyyMMddHHmm"));

string productKey = string.Format("{0}{1}", productId, timestamp);

string passPhrase = ConfigurationManager.AppSettings["PassKey"];

var encryptionBLL = new BLLEncryption();

string encryptProductKey = encryptionBLL.Encrypt(passPhrase, productKey);

using (var client = new ProductAPIServiceClient())

{

using (new OperationContextScope(client.InnerChannel))

{

var messageHeader = new MessageHeader<string>(encryptProductKey);

MessageHeader untypedMessageHeader = messageHeader.GetUntypedHeader("ProductKey", "");

OperationContext.Current.OutgoingMessageHeaders.Add(untypedMessageHeader);

//Call the ProductAPIService method like other WCF Service

}

client.Close();

}

# 3.2 Classic aspService Information

This section is to provide a brief description for the methods found in the aspService.aspx that the Classic asp application like CCNhub 1.0 can called. This is an aspx page for Classic asp application to POST the Request to CP for integration.

For the Classic asp application to call the aspService methods, it will need to post to this page URL <WebServerURL>/Service/aspService.aspx. For the <WebServerURL> refer to the next section for URL information.

When post to the aspService.aspx, it will need to pass the below Request parameters for the method call. To call the different method, it will need to post the Request for the different **Request.Params[“Page”]** value as shown in table below.

|  |  |  |
| --- | --- | --- |
| **Method Name**  **(Request Page value)** | **Method Parameters**  **(Request Params value)** | **Remarks** |
| DeleteUserAccount | ["AccountIDList"]  ["ConfigBy"] | AccountIDList is “,” separated string of accountID to delete.  ConfigBy is string for identify user calling this request. |
| CreateUserAccount | ["FirstName"]  ["LastName"]  ["UserEmail1"]  ["UserEmail2"]  ["Gender"]  ["ContactNo"]  ["UserID"]  ["City"]  ["UserType"]  ["GlobalCompanyID"]  ["ConfigBy"] | FirstName and LastName is string of user name.  UserEmail1 and UserEmail2 is string of user own email address.  Gender is string value either “M” / “F”  ContactNo is string value < 25 characters  UserID is string value < 15 characters  City is 3 characters code  UserType is string value either “A” / “U” (Admin / User)  GlobalCompanyID is Guid string of the user account’s Company its belong to.  ConfigBy is string for identify user calling this request. |
| ProductEndLog | ["SessionID"]  ["ProductID"] | SessionID is Guid string of the session to End  ProductID is string of the Product ID |
| IsSessionActive | ["SessionID"]  ["ProductID"] | SessionID is Guid string of the session to check whether it is alive  ProductID is string of the Product ID |
| LogoutCP | ["SessionID"] | SessionID is Guid string of the session to Logout |
| UpdatePassword | ["AccountID"]  ["Password"] | AccountID is Guid string of the user account  Password is string of the new password to change |
| SaveUserAccount | ["AccountID"]  ["FirstName"]  ["LastName"]  ["UserEmail1"]  ["UserEmail2"]  ["Gender"]  ["ContactNo"]  ["CompanyName"]  ["ConfigBy"] | AccountID is Guid string of the user account  FirstName and LastName is string of user name.  UserEmail1 and UserEmail2 is string of user own email address.  Gender is string value either “M” / “F”  ContactNo is string value < 25 characters  CompanyName is optional string value  Can set as empty string if no new value needed to change for the SaveUserAccount |
| SaveProductPIMA | ["CompanyProductID"]  ["ProductID"]  ["AllowedOrigin"]  ["PIMA"]  ["ConfigBy"] | CompanyProductID is long string value for the Key to update.  ProductID is string of the Product ID  AllowedOrigin is string of “,” separated City Code.  PIMA is string of PIMA address to save  ConfigBy is string for identify user calling this request. |

# Serivces URL Information

|  |  |  |
| --- | --- | --- |
| **Development** | | |
| **WebServerURL : http://devpilot.ccnhub.com/CP\_Root/** | | |
| **Web Site** | **Server** | **DEVCHBCPS1** |
| **ProductService URL** | [**http://devchbcps1epouchservices.dev.ccn/AccountService/ProductService.svc**](http://devchbcps1epouchservices.dev.ccn/AccountService/ProductService.svc) | |
| **HierarchyService** | [**http://devchbcps1epouchservices.dev.ccn/AccountService/HierarchyService.svc**](http://devchbcps1epouchservices.dev.ccn/AccountService/HierarchyService.svc) | |
| **ProductAPIService URL** | [**http://devchbcps1epouchservices.dev.ccn/ProductAPIService/ProductAPIService.svc**](http://devchbcps1epouchservices.dev.ccn/ProductAPIService/ProductAPIService.svc) | |
| **SecureProductAPI Service URL** | [**http://bizservice.dev.ccn/SecureProductAPIService/v1/SecureProductAPIService.svc**](http://bizservice.dev.ccn/SecureProductAPIService/v1/SecureProductAPIService.svc) | |
| **Pre-production** | | |
| **WebServerURL : http://ppdpilot.ccnhub.com/CP\_Root/** | | |
| **Web Site** | **Server** | **PPDCHBCPS1** |
| **ProductService URL** | [**http://ppdchbcps1epouchservices.ppd.ccn/AccountService/ProductService.svc**](http://ppdchbcps1epouchservices.ppd.ccn/AccountService/ProductService.svc) | |
| **HierarchyService** | [**http://ppdchbcps1epouchservices.ppd.ccn/AccountService/HierarchyService.svc**](http://ppdchbcps1epouchservices.ppd.ccn/AccountService/HierarchyService.svc) | |
| **ProductAPIService URL** | [**http://ppdchbcps1epouchservices.ppd.ccn/ProductAPIService/ProductAPIService.svc**](http://ppdchbcps1epouchservices.ppd.ccn/ProductAPIService/ProductAPIService.svc) | |
| **SecureProductAPI Service URL** | [**http://bizservice.ppd.ccn/SecureProductAPIService/v1/SecureProductAPIService.svc**](http://bizservice.ppd.ccn/SecureProductAPIService/v1/SecureProductAPIService.svc) | |
| **Production** | | |
| **WebServerURL : http://pilot.ccnhub.com/CP\_Root/** | | |
| **Web Site** | **Server** | **HQCHBWEB5/HQCHBWEB6** |
| **ProductService URL** | [**http://epouchservices.ccn/AccountService/ProductService.svc**](http://epouchservices.ccn/AccountService/ProductService.svc) | |
| **HierarchyService** | [**http://epouchservices.ccn/AccountService/HierarchyService.svc**](http://epouchservices.ccn/AccountService/HierarchyService.svc) | |
| **ProductAPIService URL** | [**http://epouchservices.ccn/ProductAPIService/ProductAPIService.svc**](http://devchbcps1epouchservices.dev.ccn/ProductAPIService/ProductAPIService.svc) | |
| **SecureProductAPI Service URL** | [**http://bizservice.ccn/SecureProductAPIService/v1/SecureProductAPIService.svc**](http://bizservice.ccn/SecureProductAPIService/v1/SecureProductAPIService.svc) | |

# CP Application Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Service Name** | **Description** | **Database** | **Service Type** |
| **Account Service** | This is to primary support methods related to Product and GlobalAccount in the database that other client application need for Login and their accounts.  To provide the IProductService interface for other WCF client to activate, suspend, unsuspend, terminate and validate access for the product. | UserAccount | WCF service |
| **Document Service** | This is to support physical files management for other client application using document database. | Document | WCF service |
| **Shipment Processor Service** | This is to process the FSU messages for shipment status update. | ShipmentService | Window service |
| **Shipment WCF Service** | This is to provide the MAWB shipment status update for other client application. | ShipmentService | WCF service |
| **Email Service** | This is to support other client application to send email notification. | ACE\_Monitor | WCF Service |
| **Mailbox Service** | This is to create mail box for the CP user. *note: not implemented* |  | WCF Service |
| **CPUtility Service** | This is the service to provide Empty interface methods for IProduct and IHierarchy. The CPProductService also implement IProductService interface for CP Product. | ePouch | WCF Service |
| **ProductAPI Service** | This is to provide method to log Product access in the ProductUsageLog and check IsSessionActive. | UserAccount | WCF Service |
| **CPMessageManager** | This is the service for processing the eFreight documents for the CP eFreight Channel. | ePouch | WCF Service |
| **CPSecurity Service** | This is to support BPM service to get the eFreight document and user company ID. | ePouch | WCF Service |
| **CCNhub Service** | This is to provide CCNhub user profile to CP and CCNhub related method. It also implements the IProductService interface for the CCNhub product. | ACE\_CCNP2 | WCF Service |
| **BPMMilestones Service** | Service for view workflow status and search efrieght documents. |  | Web Service |