



Interfacing with Enkompass

By S. Eric Ellis

Disclaimer

All trademarks used herein are the sole property of their respective owners.

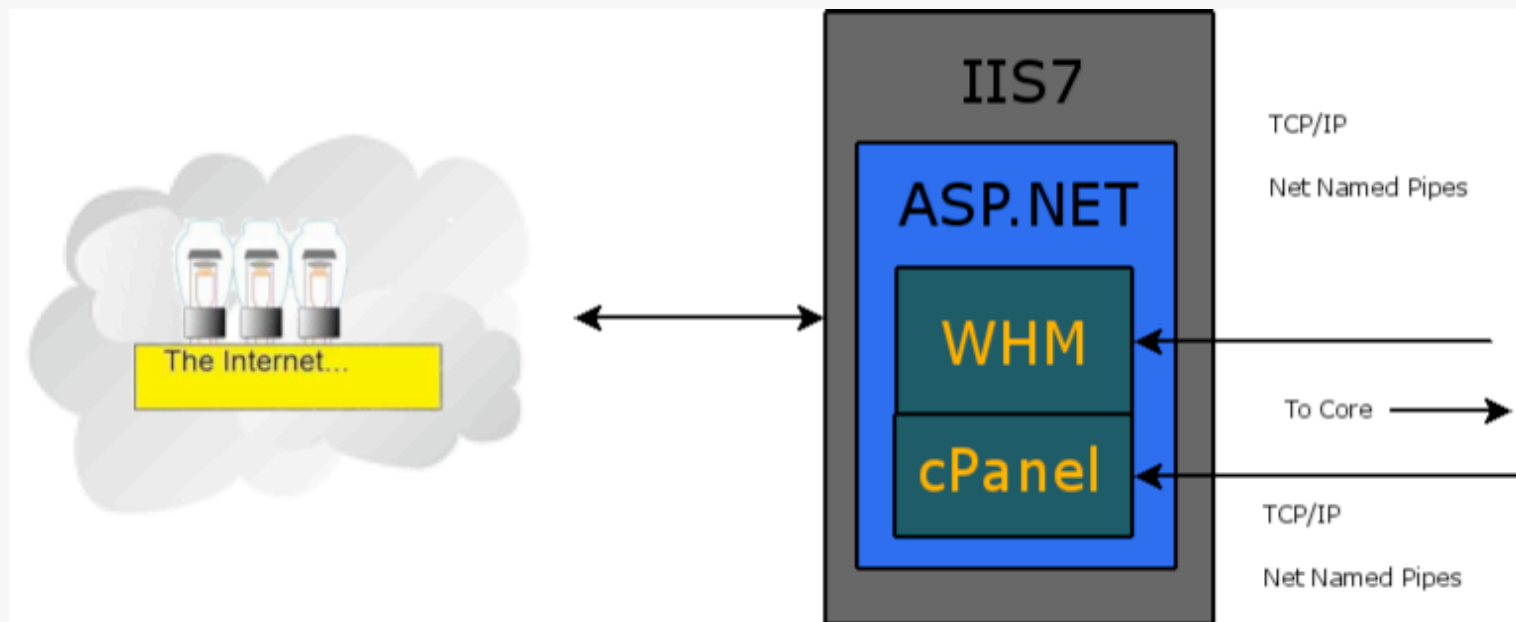
Purpose

- Why we are here today...
 - “Smarter than what you're working on”
- Enkompass outside and in
 - Interface
 - Interfacing Methods
 - Core
- Why would I want to develop for Enkompass?
- Build an example program in Visual Studio Express
- Use our example program to create a site
- Questions

Web Interface

- Copy of IIS7 Hosted Core
 - ASP.NET
 - cPanel/WHM Web Interface
- Redesigned for ASP.NET but looks similar to our existing Linux product.
 - Less of a need to retrain existing work force
- Compiled ASP.NET code will provide quick page loads.
 - Compiled on use
 - Saves space
 - Makes pages you use often faster

Interface



What is WCF

- Windows Communication Foundation
- Part of the ASP.NET framework
- A remoting API that for windows
 - Service oriented architecture
 - Allows for distributed computing
 - Allows asynchronous communication
- WCF uses endpoints with contracts to do specific functions
- Works using the ABC method
 - Address
 - Binding
 - Contract

What are Contracts

- Contracts are exposed by the endpoint
 - ServerAdministratorInterface
 - WebSiteOwnerInterface
- In most cases the contracts are the ability to do one thing on a page in either Server Administrator Interface or WebSite Owner
 - Create an account is one contract
 - Terminate an account is another contract
- Strict and limited use for each function
- We can always add more contracts for specific needs
- Make use data objects
- Can accept values such as integers strings as defined in the contract

Enkompass WCF Service

- A named windows service
- Communicates over
 - Net Named Pipes
 - TCP/IP
 - You can use either one or but not both
- Used to communicate between the core and any interface
- WCF Impersonates the user that has logged in
 - Inherits their limits and permissions
 - enforces the security
- WCF is the gatekeeper

User Control

- User limits and restrictions are stored in AD
 - It's there so let's use it
 - Plans and Features lists are stored within AD
 - Must use a Read/Write Domain Controller
 - Otherwise you won't be able to add accounts
- Trusts can be used but all cPanel users must reside in the same domain
 - Great for admin users so they don't need cPanel accounts
- Using the Windows 2003 domain scheme but 2008 is supported.

Core

- Hosts the Enkompass WCF Service
 - Which hosts our contracts to perform tasks
 - Can interface with any other third party interface written to use the contracts
- WCF allows communication between services over standard network protocols.
 - Allows us a single point to control all servers in a Enkompass Domain
- WCF is made of a “*strange kind of awesome*”
 - Very very functional
 - Needs to be told to use said features

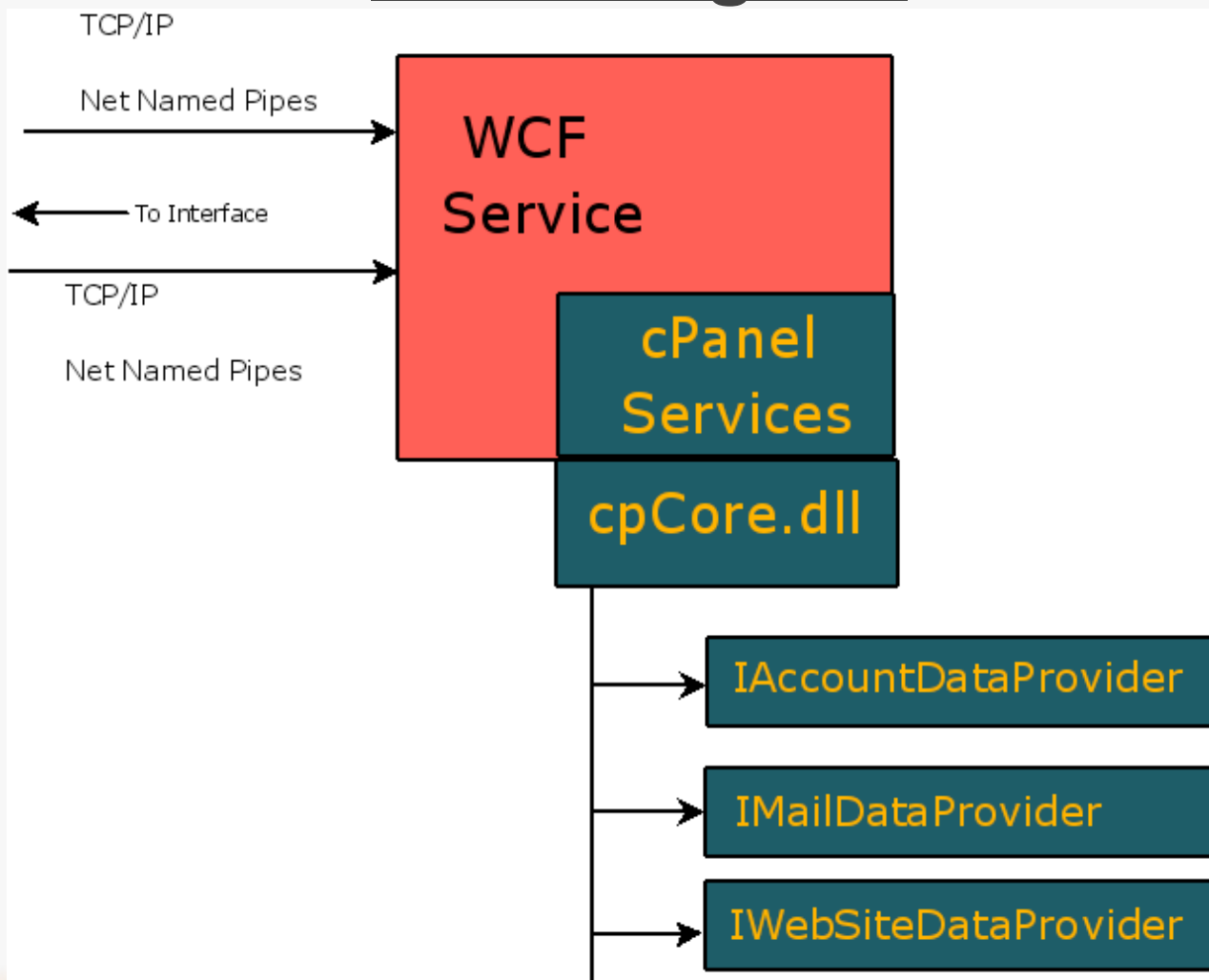
cPCore.dll

- Hosts the Providers
- Providers:
 - IAccountDataProvider
 - IMailDataProvider
 - IWebsiteDataProvider
 - IFtpSiteDataProvider
 - IDnsDataProvider
 - IDatabaseDataProvider
- This is what does all the work
- This list can grow as we need it to

cPCore.dll

- Easy naming scheme
 - Avoids confusion
 - No need to make this hard
- Modular providers
 - We can add or remove features to each provider
 - Plenty of room for growth
 - All we'll need to do is update and replace the .dll
- Example:
 - IIS8 is released
 - MySQL 6
- We could support the new version or the old one to allow you time to update at your leisure.

Core Diagram





Why would I want to program for Enkompass?

- Examples
 - Billing software
 - Customer login portal
 - Employee login portal
 - Overlord panel for multiple Enkompass domains
- Anything else you can imagine
 - Built on potential



Billing Software

- Make it easy for your Customer Service team and yourself
- Add support to create accounts on new customer entry
- Add support to suspend and unsuspend if accounts are overdue
- Add support to terminate accounts if they have been suspended for more than 90 days



Customer Login Portal

- The ultimate in branding. You could provide your customers with a login portal that has some or all of the features of WSO or SAI within an existing interface you already provide
- This will also provide you a place to serve other information:
 - Billing
 - Network
 - Support
- Built it how ever you need it for your business



Employee Login Portal

- The same could be done for a staff portal
 - Staff can work on a server without having to login into Enkompass
 - Limit access to important servers
- Limit your staff to avoid account termination of live accounts by mistake with a termination queue
 - Give a buffer of 72 hours for accounts you wish to remove
 - Provide a safety net for your employees for honest mistakes that can happen

The Overlord Panel

- For power-users and trusted users only
 - For people with the power and responsibility to get things done quickly
- Control multiple domains that have Enkompass installed
- Could be paired with the existing API to control Linux and FreeBSD cPanel servers
- Could be built as an MSC snap-in
- You could write remote programs to emulate the /scripts folder path

Before we Build

- A word about Visual Studio Express 2008
 - It's free and does everything we do here and more
 - <http://www.microsoft.com/express/>
 - It's very easy to use
 - I love vi(m), but intellisense makes things easy



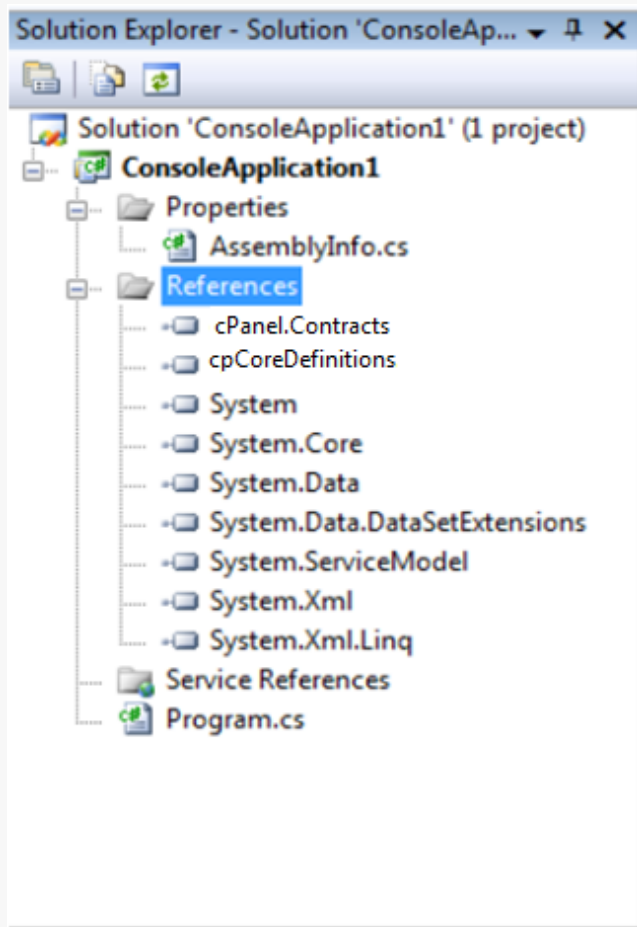
Accessing the cPCore

- Previous versions of Enkompass required you to access the cPCore.dll directly, we no longer support or recommend this method.
- We recommend using cpanelcontracts.dll with cPCoreDefinitions.dll
 - No need for Enkompass licensing on developmental machines
 - Make use of the same documentation that the Enkmopass team uses to write the product.

Building a Sample Program

- We will need to add References
- Add references to cpanel.contracts.dll and cpCoreDefinitions.dll
- Add a Reference to “System.ServiceModel” and browse to the locations of two dll files

Things Should End up Looking Like This:



- Note the References we've added
 - cpanel.contracts
 - cpCoreDefinitions
 - System.ServiceModel
- You are free to add anything else you'd like this is the bare minimum we need aside from the defaults

Priming the Pump

- Before we write any code we'll need a few lines to let C# know what we're doing
- These are related to the references we imported
- You can add anything else you may need
- Things should look like this:
 - `using System;`
 - `using System.Collections.Generic;`
 - `using System.Linq;`
 - `using System.Text;`
 - `using System.ServiceModel; //Added 4 here that we need`
 - `using System.Net;`
 - `using cpanel.contracts;`
 - `using cpCoreDefinitions;`

Getting Connected

- Once we've got a namespace we need to connect to the endpoint
- Port 2098 for SAI; 2097 for WSO
- Had we wanted to use WSO we'd use / WebSiteOwnerInterface DNS could be used here but we're keeping it simple
- User and Password don't need to be strings

```
string SaiEndPoint = "net.tcp://192.168.99.24:2098/ServerAdministratorInterface";  
// Server we're working with  
string User = "Admin";  
string Password = "AdminPassword";  
NetTcpBinding NetTCPBinding = new NetTcpBinding(SecurityMode.Transport);  
// Turning on Security  
EndpointAddress SAIEndPointAddress = new EndpointAddress(SaiEndPoint);  
// where to go in WCF
```

Opening a Channel

- Now that we know where to go and what to login with, we need to establish a connection with WCF
- WCF uses endpoints to do this
 - Endpoints are just URI addresses we'll reference
- That will look something like this:

```
ChannelFactory<IServerAdministratorService> channelfactory = new
ChannelFactory<IServerAdministratorService>(NetTCPBinding,
SAIEndPointAddress);
//This will get us in to WCF to use it
channelfactory.Credentials.Windows.ClientCredential = new
System.Net.NetworkCredential(User, Password); //Login Info
//Create the Channel
IServerAdministratorService SAIProxy =
channelfactory.CreateChannel();
```

New User Information

- Now that we have a channel open let's get the data we need to create a new account by creating an object to send over

```
AccountData AccountInfoObject = new AccountData();
AccountInfoObject.Name = "TestUser";
//User name has to 8 characters or less to be compatible with *nix users
AccountInfoObject.OwnerName = "Admin"; // the "root" user
AccountInfoObject.ContactAddress = "erice@cpanel.net";
AccountInfoObject.PrimaryDomain = "enkompass.tld";
AccountInfoObject.Language = "English";
AccountInfoObject.IP = IPAddress.Any;
AccountLimits Limits = new AccountLimits();
Limits.MaxParkedDomains = 5;
AccountInfoObject.Limits = Limits;
```

Doing the Work

- We now have all the data we need to create the object and tell WCF to build an account
- Our last line will actually make the account as such
- To do this we need the user object we made in this case “AccountInfoObject” and a password for that user

```
WHMProxy.CreateAccount(AccountInfoObject, "Pa$$w0rd!");
```

- Must be a “strong password” by default:
 - 8 characters or longer
 - Must have at least one of each of the following
 - Letter
 - Number
 - Symbol

Bonus Round

- Knowing you've created an account is great but let's get a quick list of the accounts so we can look to see if ours is there. We do so like this:

```
int outNumberOfRecords = 0;
int outNumberOfPages = 0;
AccountData[] ListUsers = SAIProxy.GetPagedAccountData( "Test*",
FieldFilter.AccountName, SortByField.AccountName, SortOrder.Ascending,
AccountInformationFilter.General, 20, 1,
out outNumberOfPages, out outNumberOfRecords, true, UserType.Unprivileged );
foreach ( AccountData user in ListUsers )
{
Console.WriteLine( "user: {0} owner: {1} domain: {2}", user.Name, user.OwnerName,
user.PrimaryDomain );
}
```

```
Console.ReadLine();
// We do this to keep it from vanishing once it's done
```