e-Business & Contacts center Solutions

Hermes.Net 4

INTÉGRATION C.T.I

Date de rédaction : 24/10/2008
Version : V1.10
Rédacteurs : Eric Papillier,
Etienne Venot,
Florian Ramilien,
Christophe Delaforge,
Frédéric Rouyvre

Sommaire

1. Rappels

Ce document a pour but de permettre d'utiliser les fonctionnalités de C.T.I à partir d'un environnement externe à Hermes.Net.

Ce document s'adresse à des utilisateurs Hermes.Net confirmés ayant, soient déjà utilisés la solution, soit ayant suivi une formation Hermes.Net V4.

Les compétences nécessaires à l'intégration de la solution Vocalcom Hermes.Net V4 sont les suivantes :

- Implémentation d'appel de web services coté serveur et coté client. (Dépendant de l'intégration souhaitée)
- Programmation JavaScript
- Programmation .net , C#, ou VB.net (uniquement pour implémentation coté serveur)

2. Définition des pré-requis minimum.

Pour pouvoir intégrer la téléphonie Hermes.Net au sein de votre application, les composants suivants auront du être installés via le setup d'Hermes.Net.

- Onnet 32 + HMP ou Dialogic.
- CT-proxy.
- OnData.
- Administration Hermes.Net
- Supervision Hermes.Net

Nota :

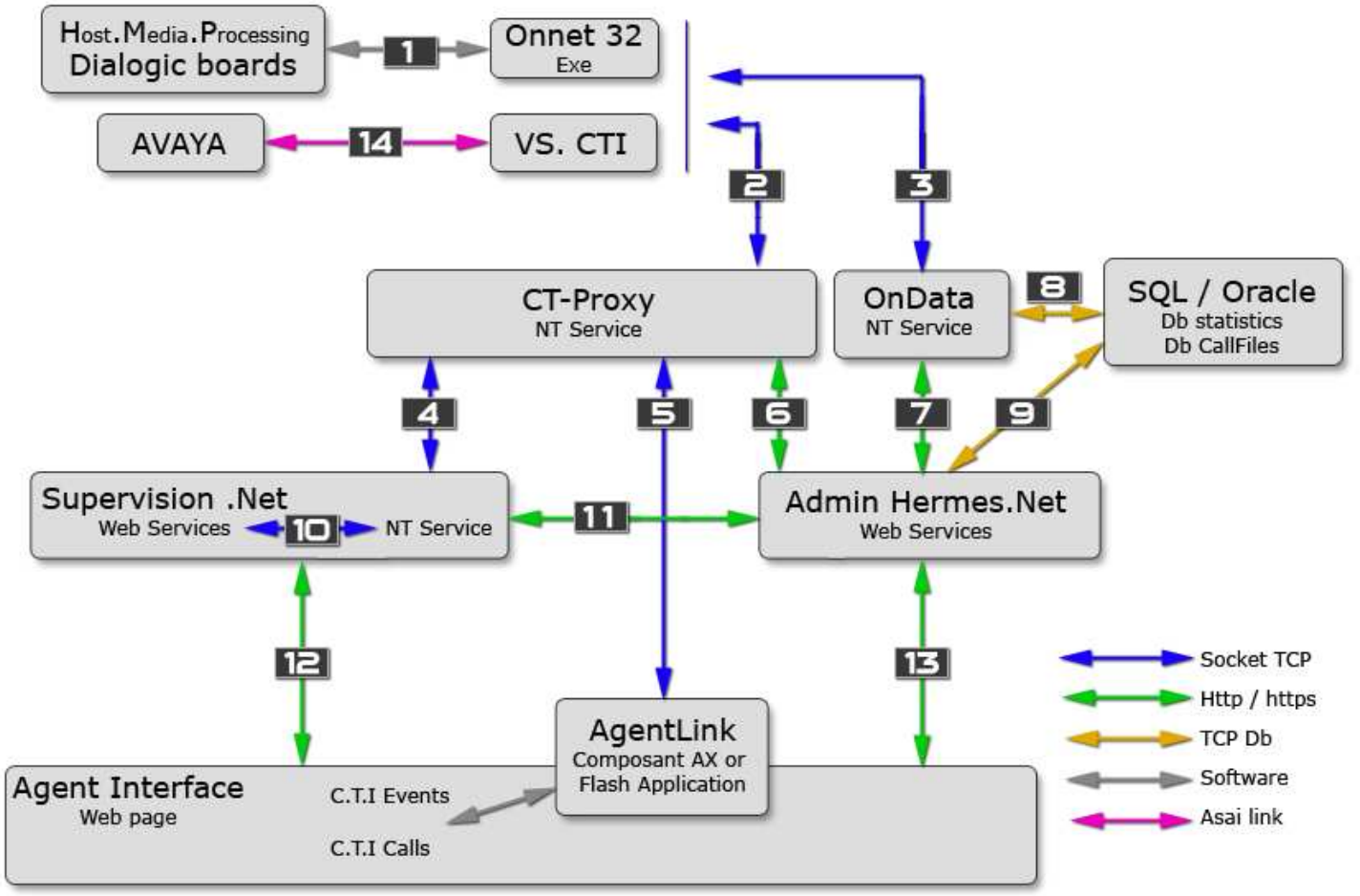
Deux solutions sont envisageables suite à l'installation des pré-requis, soit, utiliser l'interface d'administration accessible via le « launcher » Hermes.Net, soit créer ses propres configurations en appelant les méthodes des web services définies en annexe 10.

Nota :

Vocalcom préconise d'utiliser l'interface existante, et d'utiliser les web services uniquement pour l'accès aux données.

L'accès aux méthodes d'écriture doit être réservé aux fonctions simples : ajout d'agent, modification de code agent, création de station, etc.

3. Présentation de l'architecture générale.



Main C.T.I Hermes.Net's interactions

4. Liaisons Inter - modules.

1

Dialogue et pilotage HMP et/ou Dialogic

2

Onnet32 > CT-Proxy

Envoi des évènements de téléphonies (appel abouti, appel en échec, etc..)
Envoi des statistiques temps réels (appels en attentes, appel en ligne, etc..)
Envoi de messages généraux (fin de campagne, etc)

CT-Proxy > Onnet32

Envoi des actions agents (connexion, statuts, pause, etc..)
Envoi des paramètres agents.

3

Onnet32 > Ondata

Envoi des évènements de téléphonies (appel reçu, appel en échec, etc..). Utilisation de la table ODCalls.
Envoi des actions agents (connexion, raccroché, pause, etc.). Utilisation de la table ODActions.
Demande d'exécution de requêtes SQL, notamment sur les fichiers d'appels.
Demande de la configuration Admin (campagnes).

4

CT-Proxy => Supervision

Réception des informations de supervision temps réel. (État des agents, files et campagnes)
Réception de l'état des boutons de la barre de téléphonie.
Réception des messages de chat.

Supervision => CT-Proxy

Envoi de commandes de la barre de téléphonie (écoute, enregistrement, etc..)
Envoi de statistiques de production aux agents (CA, CP)
Envoi de messages de chat

5

Échanges des informations temps réels entre l'agent et la téléphonie.

6

Check de la configuration en cours et récupération de la configuration générale C.T.I de l'admin

7

Check de la configuration en cours et récupération de la configuration générale C.T.I de l'admin

8

Les taches sur SQL serveur / oracle sont :
Écriture des tables ODCalls et ODActions en fonctions des événements reçus par Onnet32.
Exécution des requêtes sur les fichiers d'appels demandées par Onnet32.

9

Gestion des fichiers d'appels, écriture, lecture, management.

10

Récupération des données temps réels stockées par le service NT en provenance du CT-proxy.

11

Vérification de la configuration en cours et récupération de la configuration générale C.T.I de l'administration.
Récupération via l'administration de l'état des fichiers d'appels.

12

Récupération des listes suivantes :

File de téléphonie
Campagnes entrante
Campagne sortante
Agents connectés

13

Récupération des informations suivantes :

Informations générales sur la configuration agent.
Liste des campagnes actives pour un agent.
Liste des contacts pour appels externes.
Liste des relances et rappels pour l'agent.

14

Echange AVAYA VS-CTI en lien ASAI.

5. Présentation sommaire des modules.

5.1 ONNET32

Application Windows permettant la gestion des interfaces CTI. Onnet32 est capable de gérer les CTI HMP et/ou Dialogic. Onnet32 est constitué d'un noyau logiciel et de scripts permettant de personnaliser les différentes taches de téléphonie (réception d'appels, émission, IVR, etc.).

5.2 VS-CTI

Ensemble de service NT dialoguant en « remoting » et permettant entre-autre de faire l'interface entre le CT-Proxy, OnData, et les bases de données. VS-Cti est aussi chargé de l'interface Asai entre le système AVAYA et Hermes.Net

5.3 CT- Proxy

Le CT-Proxy à un rôle primordial dans la solution Hermes.Net. En effet, il est le garant de la communication (au niveau téléphonie), entre les agents et le CTI. Le CT-Proxy, est une sorte de passerelle pour les événements de téléphonie temps réels. Le CT-Proxy, contient à son bord, un système permettant l'historisation des informations, le calcul de certaines valeurs (temps cumulé de pause). De part sa connexion aux web services de l'administration, il est capable de fournir à l'interface client des informations de configurations comme : les statuts, les campagnes affectées aux agents, les code de pause, etc.

5.4 OnData

Schématiquement on peut imaginer 3 grandes phases de fonctionnement d'OnData : la première étant de fournir à Onnet32 la configuration du système, la deuxième, de fournir à Onnet32 les fiches que ce dernier lui demande pour les campagnes sortantes, la troisième, de recevoir d'Onnet32 tous les événements de téléphonie et de les écrire en base de données pour l'édition des rapports.

5.5 Supervision.Net

Cette application est formée de 3 parties : une interface graphique permettant aux superviseurs de contrôler toute la production d'un plateau, d'un service NT chargé de la collecte des informations recueillis par le CT-Proxy, et d'un Web Service capable de mettre à disposition ces informations.

5.6 Admin Hermes.Net

L'administration Hermes.Net est composée de deux parties : une partie interface graphique permettant la gestion complète de la solution, et une partie web service permettant l'accès aux informations de configuration, mais aussi, à la modification ou à l'ajout de configuration. Il est à noter que l'interface graphique de l'administration n'utilise que les web services, pour stocker, lire ou obtenir des informations. Une troisième partie de l'administration est dédié à la gestion des fichiers d'appels, la gestion des transferts des fichiers vers Onnet32.

5.7 Agent Link

Permet la communication entre le CT-Proxy et l'interface agent. Peut être, soit sous forme d'un AX signé, soit d'une application Flash dépourvue d'interface graphique. Le but de ce document est en grande partie de comprendre comment intégrer ce composant à une application externe.

6. Présentation des moyens d'accéder aux informations de configuration.

6.1 Généralités

L'Admin gère le stockage des données en base, la diffusion des modifications aux autres applications Hermes, la validation des données. Cela travers quelques Web Services SOAP. Ils sont donc utilisable dans différents langages a partir des librairies adéquate (par exemple AXIS en java, Nu Soap en PHP, natif en DotNet, …)
L'Admin permet un fonctionnement en mode ASP, donc de créer des sites correspondant a différents clients / agences. Chacun des ces sites est indépendant et isolé des autres.
Cette documentation présente le fonctionnement des différents Web Services. Pour le détail de l'API, voir la documentation de tous les objets en Annexe.

NOTE : nous notons ici « Admin » avec un « A » majuscule, l'application et non pas l'utilisateur ayant les droits d'administration.

6.2 Propriétés communes des objets

Pour gérer les données sur plusieurs sites, tous les objets ont une propriété « Oid » (Object Identifier). L'Oid est une chaine de 8 caractères uniques, quelque soit le type de l'objet et son site. Cela est nécessaire car les identifiants métier ne sont pas forcément unique : chaque site peut avoir un Agent 1000.
La plupart des objets ont aussi une propriété « Owner » qui identifie leur site sauf bien sûr, les objets n'appartenant à aucun site (liste des modules, des connections, …).
Pour finir, chaque objet stocke des informations sur son historique :

- Date de création et Oid de l'administrateur l'ayant créé (CreateAt, CreateBy).
- Date de dernière modification et Oid de l'administrateur (UpdateAt, UpdateBy).
- Date de suppression (DeleteAt, DeleteBy).

6.3 Web Service 'GetConfig'

Permet la lecture des informations de tous les objets gérés par l'Admin. Ce service est utilisable par tout le monde sans gestion d'authentification (seule une restriction sur les adresses IP est possible).

Pour chaque objet, au minimum deux méthodes sont disponibles : liste des objets pour un site, accès a un objet via son Id.

Utilisation des Changes Listeners

Le mécanisme des « changes » permet de recevoir en temps réel les modifications effectuées par les administrateurs. Cela est utile si l'on ne veut pas redemander systématiquement les listes d'agents, listes de campagnes, …

Trois méthodes sont disponibles :

- ChangesRegister : permet de s'enregistrer, renvoi un identifiant de session.
- ChangesFilter : permet de choisir les objets que l'on souhaite consulter (optionnel).
- ChangesList : renvoi toutes les modifications ayant eu lieu depuis le dernier appel.

Interfaces disponibles

Il est possible d'accéder directement au WS de l'Admin depuis le langage souhaité. Cependant certains « packages » sont disponibles pour faciliter l'accès :

- AdminConnector : DLL prenant en charge l'accès à tous les Web Services et utilisant les « Changes Listeners ».
- HermesScript : Accès aux Web Services dans des scripts Windows (script d'imports, de migration, …).
- Stub JavaScript : Fichier JavaScript contenant toutes les fonctions des Web Services.

Voir en Annexe pour ces scripts.

6.4 Web Service 'SetConfig'

Web Service permettant la lecture et la modification de tous les objets gérés par l'Admin. Ce service nécessite de s'authentifier (et donc de maintenir une session).

L'authentification et les transactions

Avant tout, l'utilisateur doit s'authentifier en utilisant la méthode « VerifyLogin ». Cette méthode prend en paramètre le login de l'administrateur et le mot de passe, encodé en SHA1. Si l'administrateur peut accéder à plusieurs sites, le choix du site se fait avec la méthode « ChangeGroup ».

Pour gérer la validité des objets, ce service utilise des transactions. Toute modification n'est sauvegardée que lors de la validation de la transaction. Les transactions utilisent un modèle d'isolation forte : un objet lut une fois pendant la transaction sera toujours renvoyé à l'identique, même si il a était modifié par un autre administrateur. Pour recharger les objets, il faut commencer une nouvelle transaction.

Les méthodes permettant de gérer les transactions sont :

- Begin : Commence une nouvelle transaction.
- Commit : Clôture une session en sauvegardant les modifications
- Rollback : Clôture une session en annulant les modifications
- Valid : Sauvegarde les modifications en cours de transaction (Identique a 'Commit' mais ne ferme pas la transaction)
- Cancel : Annule les modifications faite depuis l'ouverture de la transaction ou le dernier 'Valid'.

Accéder aux objets

En fonction du compte administrateur utilisé, le service donne accès à un seul site ou plusieurs sites/sociétés. Pour chaque objet de l'Admin, 5 méthodes au minimum sont disponibles :

- ListXxxx() : renvoie tous les objets du type 'Xxxx' du site courant
- GetXxxx(id) : renvoie l'objet demandé du site courant
- CreateXxxx(obj) : crée et renvoie l'objet mis a jour (création de L'Oid, ...)
- UpdateXxxx(obj) : met a jour l'objet
- DeleteXxxx(obj) : supprime l'objet

7. Exemple d'intégration client léger + AX.

La version AgentLink ActiveX est basé sur l'ActiveX 'Vocalcom Agent Toolbar'. AgentLink permet la connexion en socket TCP au CT-Proxy et affiche un bandeau téléphonique.
Etant donné que AgentLink maintient une socket TCP avec le CT-Proxy, si la page le contenant est rechargée, il sera nécessaire de réinitialiser la connexion et l'identification au proxy.
Nota : Les fichiers utilisés par cette page se trouvent dans le répertoire 'Intégration Bandeau ActiveX' placé à la racine de ce document.

7.1 Contenu minimum de la page web de test

Déclaration de l'application ActiveX

```
<OBJECT id="Toolbar" language="vbscript" align="middle" width="800px" height="100px"
                classid="clsid:D82F4AEA-E455-11D3-A715-204C4F4F5020"
                CODEBASE="path_to_ActiveX/AgentLink.cab#version=4,0,0,8081"
</OBJECT>
```

Remplacer « **path_to_ActiveX »** par le chemin des fichiers sur le serveur. Par défaut en V4 : « http://*NomDuServeur*/hermes_net_v4/PlateformPublication/ActiveX_Flash/ »

Initialisation et fermeture de l'ActiveX

```
<script type="text/javascript">
var agentlink_toolbar;

function init()
{
     agentlink_toolbar = document.getElementById("Toolbar");
     agentlink_toolbar.object.AgentProxy='10.0.10.105';
     agentlink_toolbar.object.Port=9992;
     agentlink_toolbar.object.Connect();
}

function close()
{
     // Le Disconnect est nécessaire, sinon AgentLink restera actif tant qu'une
fenêtre de IE sera ouverte.
     agentlink_toolbar.object.Disconnect();
}
</script>
<body onload="init();" onunload="close()"> ….
```

7.2 Explications

L'ActiveX est chargé par la balise OBJECT depuis le serveur web avec le CODEBASE et enregistré dans Windows. Si l'active X est déjà enregistré sur vos postes, le CLASSID est suffisant pour charger l'ActiveX.
Ensuite, nous déclarons un objet JavaScript de type « Toolbar » et que nous nommons « agentlink_toolbar ». Ce nom n'est pas fixé. C'est sur cet objet que vous pourrez appeler les méthodes et récupérer les paramètres CTI.
Cet objet est utilisé directement dans la méthode « Init » pour configurer l'adresse IP su serveur CT-Proxy ainsi que sont port.

7.3 Exemple d'utilisation

Dans cet exemple, nous nous basons sur les déclarations faites dans la partie précédente de ce document.
Sont notées en magenta, les variables JavaScripts « utilisateurs ».

```
<script type="text/javascript">
var agentlink_toolbar;

function init()
{
     agentlink_toolbar = document.getElementById("Toolbar");
     agentlink_toolbar.object.AgentProxy='ip_proxy';
     agentlink_toolbar.object.Port='port_proxy';

     agentlink_toolbar.attachEvent("Connect", MyConnectFunction);
     agentlink_toolbar.attachEvent("UserIdentification", MyIdentificationFunction);
     agentlink_toolbar.object.Connect();
}

function close()
{
     // Le Disconnect est nécessaire, sinon AgentLink restera actif tant qu'une
fenêtre de IE sera ouverte.
     agentlink_toolbar.object.Disconnect();
}

function MyConnectFunction()
{
     alert("AgentLink is connected to the Proxy : send identification");
     agentlink_toolbar.object.Login(id_agent, password_agent, station_agent);
}

function MyIdentificationFunction(userId, userName)
{
     if (agentlink_toolbar.object.LoggedIn != false)
           alert("Agent identifié");
     else
           alert("Erreur lors de l'identification de l'agent");
}

function makeCall()
{
     var number = document.getElementById("phonenumber");
     agentlink_toolbar.object.ManualCall(number.value);
}
</script>
```

Avec en plus dans le code HTML de la page, l'appel de makeCall :

```
<form>
     <input type="text" id="phonenumber" value="0155373050" >
     <input type="button" value="Make Call" onclick="makeCall()">
</form>
```


8. Exemple d'intégration client léger + Flash.

La version Agent Link Flash est composée d'une application Flash et d'un ensemble de fichiers JavaScript. L'application Flash permet la connexion en socket TCP au CT-Proxy, les fichiers JavaScript permettent d'exposer les mêmes méthodes et propriétés que l'AX Agent Link. Il ne sera donc pas compliqué d'utiliser soit la version Active X, soit la version Flash.

Les fichiers JavaScripts sont fournis encodés. Afin d'être transmis à la page web décodée, ils doivent être placés dans un des répertoires publiés de l'administration.

L'application Flash et les fichiers JavaScript doivent être inclus dans la même page.
Etant donné que le Flash maintient une socket TCP avec CT-Proxy, si la page le contenant est rechargée, il sera nécessaire de réinitialiser la connexion et l'identification à CT-Proxy.

Nota : ***Les fichiers utilisés par cette page se trouvent dans le répertoire 'Intégration Bandeau Agent Flash/' placé à la racine de ce document.***

8.1 Contenu minimum de la page web de test

Déclaration des fichiers JS génériques à inclure.

```
<script language="javascript" src="path_to_Flash/core/Toolbar.ashx"></script>
<script language="javascript" src="path_to_Flash/core/AgentLink.ashx"></script>
<script language="javascript" src="path_to_Flash/core/AgentButton.ashx"></script>
<script language="javascript" src="path_to_Flash/core/CallStatus.ashx"></script>
<script language="javascript" src="path_to_Flash/core/CallStatusGroup.ashx"></script>
<script language="javascript" src="path_to_Flash/core/PauseCode.ashx"></script>
<script language="javascript" src="path_to_Flash/core/CallInformation.ashx"></script>
<script language="javascript" src="path_to_Flash/core/Accessories.ashx"></script>
<script language="javascript" src="path_to_Flash/core/Status.ashx"></script>
<script language="javascript" src="path_to_Flash/core/SearchModeInformation.ashx"></script >
<script language="javascript" src="path_to_Flash/core/ProxySupport.ashx"></script>
```

Déclaration du fichier JS permettant l'intégration facilitée de l'agent Flash.

```
<script language="javascript" src="path_to_Flash/swfobject.js"></script>
```

Déclaration de l'application Flash

```
<div id="Flash" style="position:absolute;top:50px;visibility:hidden"></div>

<script language="javascript">
var so = new SWFObject("path_to_Flash/AgentLink.swf", "AgentLink", "100", "30", "8", "");
so.write(document.getElementById("Flash"));
agentlink_toolbar = new Toolbar("Flash");
```



```
</script>
```

Remplacer “ **path_to_Flash** ” par le chemin des fichiers sur le serveur.

8.2 Explications

Nous plaçons l'application Flash dans un div masqué. L'id de ce div n'est pas fixé, il est donc possible de le modifier si nécessaire.

L'objet “SWFObject” va créer le code html nécessaire à l'inclusion du flash dans la page. Le deuxième paramètre passé à la fonction indique avec quel ID il sera créé.

Il est nécessaire de toujours passer “AgentLink”, c'est en effet ainsi que les méthodes javascripts sauront appeler les méthodes de l'application Flash.

Enfin, nous déclarons un objet javascript de type “Toolbar” et que nous nommons “agentlink_toolbar”. Ce nom n'est pas fixé. C'est sur cet objet que vous pourrez appeler les méthodes et récupérer les paramètres CTI.

8.3 Exemple d'utilisation

Dans cet exemple, nous nous basons sur les déclarations faites dans la partie précédente de ce document.

Sont notées en magenta, les variables JavaScripts « utilisateurs ».

```
function MyConnectFunction()
{
// AgentLink is connected to the Proxy : send identification
agentlink_toolbar.object.Login(id_agent, password_agent, station_agent);
}

function MyIdentificationFunction(userId, userName)
{
if (agentlink_toolbar.object.LoggedIn != false)
{
alert(“Agent identifié”);
}
else
{
alert(“Erreur lors de l'identification de l'agent”);
}
}

// attachement de l'évènement cti “Connect” à la fonction personnalisée
“MyConnectFunction”
agentlink_toolbar.attachEvent("Connect", MyConnectFunction);

// attachement de l'évènement cti “UserIdentification” à la fonction personnalisée
“MyIdentificationFunction”.
agentlink_toolbar.attachEvent("UserIdentification", MyIdentificationFunction);

// Envoi de la demande de connexion au proxy
agentlink_toolbar.object.AgentProxy='ip_proxy';
agentlink_toolbar.object.Port='port_proxy';
var connected = agentlink_toolbar.object.connect();

if (!connected)
{
```



```
alert("Erreur lors de la connexion au proxy");
}
```

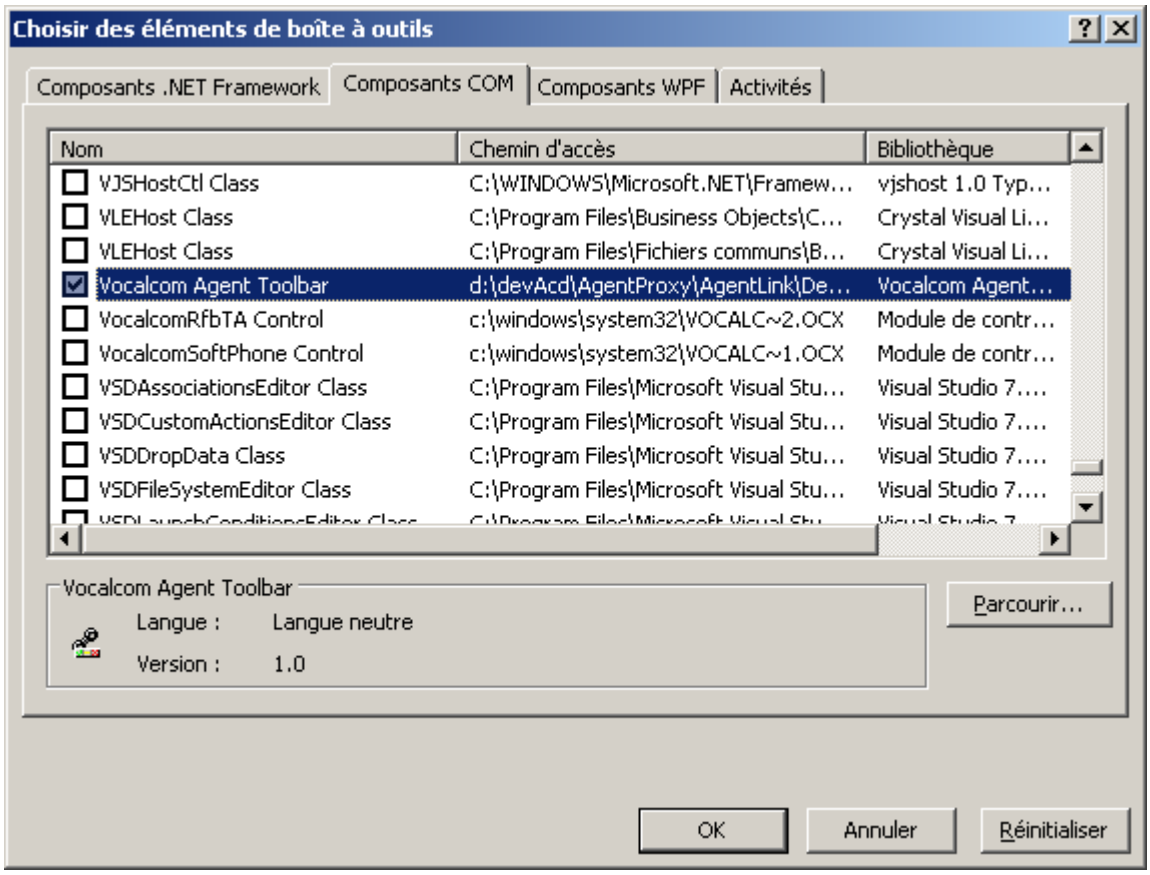

9. Exemple d'intégration client lourd.

Exemple d'intégration d'AgentLink AX dans un client Windows Forms en C#. Cet exemple montre une intégration minimale permettant le fonctionnement du bandeau. Un exemple plus complet se trouve en annexe.

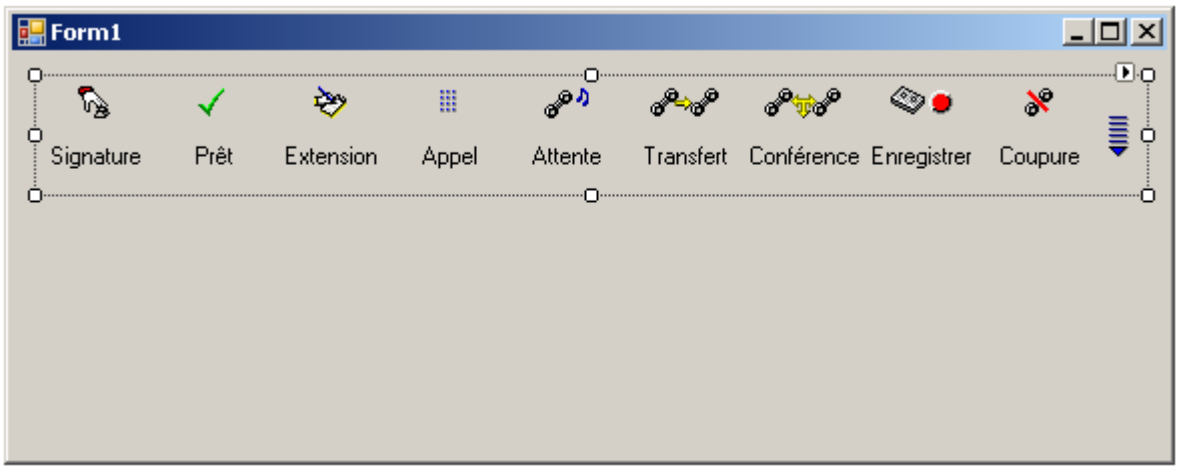
Nota : ***Cet exemple se trouve dans le répertoire « Exemple AgentLink Client EXE /» placé à la racine du document***

9.1 Intégration de l'ActiveX

Ajouter le composant AgentLink dans la boite a outils de Visual Studio. Si AgentLink a été installé sur ce poste (lors de l'utilisation du bandeau agent par exemple), il est alors enregistré par défaut dans 'c:\windows\system32' et il est aussi enregistré comme composant COM « Vocalcom Agent Toolbar ». Dans 'Boites a outils', sélectionner 'choisir les éléments …' et ajouter 'Vocalcom Agent Toolbar'.



AgentLink AX est prêt à être ajouté dans un formulaire. Lors de cet ajout, deux références vont être ajoutées au projet : 'AgentLink' et 'AxAgentLink'.



Pour qu'AgentLink soit utilisable, il ne reste plus qu'a définir le CT-Proxy sur lequel il doit se connecter. Pour cela, sur le chargement de la Form, initialiser le port et l'ip et appeler la méthode « Connect ».

```
private void Form1_Load(object sender, EventArgs e)
{
     axToolbar1.AgentProxy = "10.0.10.105";
     axToolbar1.Port = "9992";
     axToolbar1.Connect();
}
```

AgentLink est maintenant initialisé et utilisable pour connecter un agent. Pour cela, cliquer sur « Signature ». Sinon, la méthode permettant de connecter l'agent est « Login(Id, Password, Station) ».
De manière à diagnostiquer les différents problèmes qui peuvent être rencontrés, AgentLink AX génère une trace dans le dossier : « C:\Documents and Settings*WindowsUser*\Local Settings\Temp\AgentLink.log ». Ce niveau de trace peut être augmenté en mettant à « True » le paramètre « Trace ».

9.2 Autres méthodes et événements

Toutes les actions possibles via les boutons d'AgentLink AX le sont aussi via des méthodes de l'ActiveX. Toutes les méthodes et les événements sont décrits dans un autre document en annexe, ci-dessous quelques exemples.

Trois événements permettant de suivre l'état de l'agent :

- ConnectEvent : Connexion acceptée par le proxy
- DisconnectEvent : Déconnexion, peut provenir de la demande du client, de CT-Proxy, de Onnet, d'un problème réseau, ...
- ButtonStateChange : Appelé a chaque changement d'état de l'agent. Permet de connaître sont état.

Exemple :

```
private void axToolbar1_DisconnectEvent(object sender, EventArgs e)
{
     MessageBox.Show("Disconnected from proxy");
}
```



```
private void axToolbar1_ConnectEvent(object sender, EventArgs e)
{
     MessageBox.Show("Connected to proxy");
}

private void axToolbar1_Info(object sender, AxAgentLink._IToolbarEvents_InfoEvent
e)
{
     MessageBox.Show(e.agentState);
}
```

La methode 'ManualCall' permet de passer un appel depuis l'application cliente.

```
private void Call_Click(object sender, EventArgs e)
{
     axToolbar1.ManualCall("0155373050");
}
```


10. Présentation des autres fonctionnalités de la barre agent Hermes.Net (hors AgentLink)

10.1 Récupération des informations agent.

Permet au WorkSpace agent (interface principale de l'agent Hermes.Net V4) de connaître la totalité de la configuration de l'agent.

Appel

Application : **Administration**
Web Service : **InfoAdmin.asmx**
Méthode : **GetFramesetParams**

Informations retournées

Le code de l'agent
Le mot de passe de l'agent
Le CT-Proxy à utiliser (IP et Port)
Le serveur rfb à utiliser
Le serveur MediaServer à utiliser
Le webservice Supervision à utiliser (pour côté client et côté serveur)
Le webservice Carnet d'Adresse à utiliser (pour côté client et côté serveur)
Booléen indiquant si la station correspond à une station sip avec softphone intégré
Ip du SipProxy (vide si non utilisé)
Login sur le SipProxy (vide si non utilisé)
Mot de passe sur le SipProxy (vide si non utilisé)
Id du site
Booléen indiquant si l'interface agent doit être modale
Le webservice FlashMediaServer à utiliser (pour côté client)

10.2 Récupération de la liste des campagnes actives.

Permet d'afficher la liste des campagnes de l'agent, et de proposer la campagne sortante à utiliser s'il y en a plusieurs.
Nota : Si l'agent a plusieurs campagnes sortantes de définies, pour qu'il puisse travailler en émission d'appel, il doit obligatoirement spécifier l'id de la campagne à utiliser lors de son appel à la méthode SetReady de AgentLink.

Appel

Application : **Administration**
Web Service : **GetInfoSupervision.asmx**
Méthode : **GetAllCampaignsFor**

10.3 Gestion des appels manuels / transferts.

Permet de récupérer la liste des contacts définis dans l'administration.

Appel

Application : **Administration**
Web Service : **GetInfoSupervision.asmx**
Méthode : **GetContactsForCustomer**

10.4 Récupération des files d'attente de téléphonie.

Appel

Application : **Supervision**
Web Service : **ReadInfo.asmx**
Méthode : **GetInboundQueueList**

10.5 Récupération des campagnes entrantes.

Appel

Application : **Supervision**
Web Service : **ReadInfo.asmx**
Méthode : **GetInboundCampaignList**

10.6 Récupération des campagnes sortantes.

Appel

Application : **Supervision**
Web Service : **ReadInfo.asmx**
Méthode : **GetOutboundCampaignList**

10.7 Récupération de la liste des autres agents connectés.

Renvoie la liste des agents connectés. Suivant les paramètres passés, cette méthode peut retourner la liste incluant les superviseurs et/ou les agents uniquement du même groupe de supervision.

Appel

Application : **Supervision**
Web Service : **ReadInfo.asmx**
Méthode : **GetIAgentList**

10.8 Affichage des informations de relances et rappels.

a - Affichage de la liste des rappels pour un agent

Appel

Application : **Admin/datamanager**
Web Service : **callback.asmx**
Méthode : **GetCallBack**

b – Récupération du nombre de rappels et relances d'un agent sur une campagne.

Appel

Application : **Admin/datamanager**
Web Service : **callback.asmx**
Méthode : **GetCallBackByCampaign**

c - Affichage du nombre des prochains rappels et relances d'un agent sur une campagne.

Appel

Application : **Admin/datamanager**
Web Service : **callback.asmx**
Méthode : **GetNextCallBacksByCampaigns**

11. Exemple d'accès aux données d'administration.

Nota : ***Un exemple C# 2 est donné dans le répertoire « Exemple Acces au Web Service Admin/ » .***

11.1 Exemple basic d'utilisation GetConfig [C# 2.0]

```
// Créer une Web Référence sur "admin/Web_Service/GetConfig.asmx", puis :
ReadTools getConfig = new ReadTools();

// Renvoi tous les agents du site 1 :
Agent[] agts = getConfig.ListAgent("1");
foreach (GetConfigWS.Agent agt in agts)
      System.Console.WriteLine(agt.AgentId + "-" + agt.FirstName);

// Renvoi l'agent 1000 du site 1
Agent agent = getConfig.GetAgent(1000, "1");
```

11.2 Exemple basic d'utilisation des 'changes' [C# 2.0]

```
// Créer une Web Référence sur "admin/Web_Service/GetConfig.asmx", puis :
ReadTools getConfig = new ReadTools();

// Enregistrement pour le site 1 (chaine vide pour tous les sites)
string changesId = getConfig.ChangesRegister("1");

// On ne demande que les modifications sur les agents (optionnel)
getConfig.ChangesFilter(changesId, new string[] { "Agent" } );

// Initialisation de nos listes
Dictionary<int, Agent> agts = new Dictionary<int, Agent>();
foreach (GetConfigWS.Agent agt in getConfig.ListAgent("1"))
  agts.Add(agt.AgentId, agt);

// On boucle sur les traitements a effectué et la gestion des mises a jour
while (true)
{
      // TODO : mettre ici le code de l'application
      // ...

      // On vérifie les modifications
      foreach (Change change in getConfig.ChangesList(changesId))
      {
            switch (change.Operation)
            {
                  case GetConfigWS.ChangeOperation.Create:
                  case GetConfigWS.ChangeOperation.Update:
                        agts[(int)change.Id] = (Agent)change.Obj;
                        break;
                  case GetConfigWS.ChangeOperation.Delete:
                        agts.Remove((int)change.Id);
                        break;
            }
      }
}
```


11.3 Exemple basic d'utilisation SetConfig [C# 2.0]

```
ServiceTools setConfig = new ServiceTools();
setConfig.CookieContainer = new System.Net.CookieContainer();

// Password transmis en hash SHA1, codé Base64.
string hash = FormsAuthentication.HashPasswordForStoringInConfigFile("jusosoxy",
"SHA1");
Authentification auth = setConfig.VerifyLogin("admin", hash);
if ( auth < Authentification.Admin )
      Console.WriteLine("Bad Login !");

// Ouverture de la transaction
setConfig.Begin();

// Creation d'un nouvel agent
Agent newAgent = new Agent();
newAgent.AgentId = 9000;
newAgent.FirstName = "John";
newAgent.LastName = "Doe";

// Envoi du nouvel agent a l'admin (pas encore d'ecriture en base)
// Ne pas oublier de reprendre l'agent renvoyé !
newAgent = setConfig.AddAgent(newAgent);

// Update agent
newAgent.Password = "password";
setConfig.UpdateAgent(newAgent);

// Données ecrites en base et diffusé a tous les services.
setConfig.Commit();
```


12. Annexe 1 : Description générale des méthodes des web services Administration.

Cette partie se trouve dans le répertoire /Web Services Administration/index.html

13. Annexe 2 : Description générale des méthodes des web services de supervision.

Cette partie se trouve dans le répertoire /Web Services Supervision/index.html

Cette documentation au format HTML décrit les méthodes utilisables dans le cadre d'une intégration avec une application tierce. Elle présente en particulier le web service "ReadInfos.asmx" de la supervision qui se trouve dans le répertoire "hermes_net_v4/Supervision/Web_Service/" d'une installation standard.

Le point d'entrée de cette documentation est la description de la classe ReadInfos qui est la classe de ce web service.

Dans le cadre d'une intégration avec le bandeau agent, on peut notamment utiliser les méthodes suivantes qui ne nécessitent que la connaissance de l'oid de l'agent :

- **GetAgentList** : donne une liste d'agents connectés
- **GetOutboundCampaignList** : donne une liste de campagnes sortantes
- **GetInboundCampaignList** : donne une liste de campagnes entrantes
- **GetMailCampaignList** : donne une liste de campagnes de mail
- **GetChatCampaignList** : donne une liste de campagnes web
- **GetInboundQueueList** : donne une liste de files d'attente utilisées par des campagnes entrantes
- **GetMailQueueList** : donne une liste de files d'attente utilisées par des campagnes de mail
- **GetChatQueueList** : donne une liste de files d'attente utilisées par des campagnes web
-

Toutes ces méthodes donnent des informations de supervision temps réel et les valeurs retournées dépendent donc du moment de l'appel de la méthode.
Nous vous invitons à consulter la documentation HTML du web service pour avoir plus de renseignements sur les valeurs retournées par ces méthodes, ainsi que sur les autres méthodes disponibles.

14. Annexe 3 : Description générale des méthodes et events de l'AX AgentLink.

Nota : ***les fichiers nécessaires et d'exemple sont fournis dans le répertoire « Documentation AgentLink/ »***

15. Annexe 4 : Hermes Script : Accès aux Web Services dans des scripts windows.

Nota : ***les fichiers nécessaires et d'exemple sont fournis dans le répertoire « Hermes_Net Scripting/ »***

AVERTISSEMENTS

* Les modifications effectuées par les scripts sur la Base sont DEFINITIVES

* Les scripts peuvent inscrire en BD des objets erronés empêchant le bon fonctionnement des autres applications (WebAdmin, OnData, ...)

DOC

Pour avoir la liste des Web Services et des objets voir l'Annexe 1 de la documentation d'Intégration de Hermes.Net

Pour une première utilisation voir le fichier script/example.js qui contient des exemples d'opérations de base pouvant être effectué via les Web Services.

UTILISATION

Pour chaque application à exécuter, effectuer les étapes suivantes :

1/ Modifier le fichier common.config

-> URL du WebAdmin
-> Login / Password de l'admin
-> Id du site a lire et/ou modifier
-> Infos complémentaires nécessaires à l'exécution du script (Base de données pour l'import, ...)

2/ Exécuter le fichier Xxxx.bat. Trois erreurs peuvent se produire :

a/ Impossible de trouver l'outils jsc.exe
-> Vérifier l'installation du framework .Net sur la machine

b/ Erreur de compilation.
-> Le code du script est incorrecte, vérifier la syntaxe

c/ Erreur d'exécution
-> Le login est incorrecte, un objet existe déjà, ...
-> Pour la plupart des scripts, en cas d'erreur AUNCUNE modification ne sera apporté à la Base. Les opérations effectuées ne sont inscrites en base
que lors de l'exécution de la commande 'Xxxx.Commit();' à la fin du script.

