Microsoft

PRESTO, un protocole d'échange pour les besoins de l'administration électronique

Version 1.0
Publication : Juin 2007

Résumé

Ce livre blanc présente les défis architecturaux auxquels sont confrontées les solutions d'échange pour l'administration électronique, les domaines réclamant une attention particulière (et qui sont, dans un premier temps, largement sous-estimés), les options mises à disposition et une démarche préconisée pour venir à bout de ces difficultés. Ce document explique ensuite plus en détail l'approche retenue par le « *PRotocole d'Échanges Standard et Ouvert* » (alias PRESTO) pour mieux relever ces défis. Enfin, ce document traduit la mise en œuvre d'une plateforme de services adossée au protocole d'échange PRESTO à l'aide des produits et technologies Microsoft.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • Etats-Unis

Sommaire

1. Introduction

« 500 millions pour la comptabilité publique, 40 millions pour l'état-civil, 20 millions pour les notaires, 8 millions pour le contrôle de la légalité, etc. Plus d'un milliard de documents sont échangés entre administrations chaque année ![1] »

Les milliers de tonnes de papier que cela suppose sont progressivement remplacées par des échanges électroniques entre les différents acteurs concernés par l'Ordonnance n° 2005-1516[2] sur les échanges électroniques.

La modernisation de l'administration pour faire de l'e-administration une réalité suppose de disposer, à la base, d'une infrastructure interopérable d'échange de confiance au service de l'administration électronique.

En effet, alors que les administrations multiplient les communications entre systèmes d'information, les solutions propices à leur développement sont nombreuses : développement interne, utilisation de progiciel, choix d'un tiers de télétransmission, etc. Cependant, les outils développés sont rarement interopérables et adaptés pour supporter de nouveaux types d'échanges.

Face à ce constat et dans le contexte de l'initiative ADELE[3], la Direction Générale pour la Modernisation de l'État – Service pour le Développement de l'Administration Électronique (DGME – SDAE) a lancé un vaste chantier permettant de créer un protocole d'échange de données qui puisse répondre à la majorité des cas d'usage. Ceci suppose :

- de non seulement définir un protocole pour assurer les échanges de messages informatiques entre les systèmes, services et applications de l'administration et des partenaires affiliés (ministères, collectivités locales, sphère santé-social, différents autres partenaires rendant un service public ou assimilé) répondant aux exigences d'interopérabilité,
- et, au delà de simples observations et recommandations d'usage, de préciser l'ensemble des mécanismes nécessaires pour résoudre des problèmes de sécurité, de fiabilité, de détection d'erreur, etc.

La définition d'un tel protocole se place dans le volet technique de l'initiative 12[4] (IT12) « Gestion des processus et intégration de services » du schéma directeur de l'administration électronique[5]. L'IT 12 relative à la gestion des processus et intégration de services qui regroupe l'ensemble des éléments qui permettent aux SI des administrations de communiquer, de créer des services en ligne mutualisés, de créer des services en ligne faisant intervenir différents partenaires, etc.

Dans ce contexte, le projet « A131b - Protocole d'échange »[6] d'IT12 a vu le jour pour s'intéresser précisément à cette définition.

[1] Cf. http://www.dent.caissedesdepots.fr/commun/pdf/publications/Plaquette-fast.pdf

[2] Cf. http://www.legifrance.gouv.fr/WAspad/UnTexteDeJorf?numjo=ECOX0500286R.

[3] Cf. http://www.adele.gouv.fr.

[4] Ce volet a pour objectif d'offrir un ensemble de solutions techniques pour les échanges de données entre systèmes d'information d'administration électronique. Il met en avant les solutions existantes et permet la création de nouvelles solutions au sein de groupes de travail inter administration.

[5] Cf. http://www.thematiques.modernisation.gouv.fr/axes/50_13.html.

[6] Cf. http://synergies.modernisation.gouv.fr/rubrique.php3?id_rubrique=165.

Le 13 septembre 2006, la version 1.0 de la spécification « *PRotocole d'Échanges Standard et Ouvert* » (alias PRESTO) a été publiée. PRESTO vise à fournir une couche d'échange de message générique afin de pouvoir échanger potentiellement n'importe quels messages au sein de l'administration publique électronique.

Les fondements techniques de cette spécification et l'architecture associée doivent non seulement permettre de faciliter les échanges d'information mais également, de pouvoir par la suite, mettre en place des systèmes de services distribués facilement accessibles et intégrables. A cette fin PRESTO propose un profil de communication élaboré permettant de répondre aux exigences d'interopérabilité et de modularité (adaptation du même protocole aux cas d'utilisation), dans le respect des standards internationaux.

Une telle spécification s'inscrit dans une démarche commune à plusieurs pays et doit également servir de base aux communications avec les partenaires européens voire également être ouvert à d'autres partenaires extranationaux. L'Allemagne, la Suède, le Danemark et l'Estonie sont chacun en cours de création d'un protocole proche de PRESTO. Le protocole européen eLink ayant été abandonné, la commission européenne, dans le cadre du programme IDABC[7] (*Interoperable Delivery of Pan-European eGovernment Services to Public Administrations, Business and Citizens*), assiste ces pays dans la définition d'un profil commun s'appuyant sur PRESTO, mais basé sur des profils d'interopérabilité internationaux.

Dans ce contexte, ce livre blanc revient sur :

1. les défis architecturaux auxquels sont confrontées de telles solutions d'échange pour l'administration électronique ;
2. les orientations et axes d'architecture retenus pour la définition du protocole d'échange PRESTO pour les besoins de l'administration électronique.

et s'intéresse aux services et à l'architecture d'un intergiciel de communication mettant en œuvre ce protocole. Pour donner une dimension pratique à ces réflexions, il illustre enfin comment les produits et technologies Microsoft peuvent contribuer à mettre en œuvre tant le protocole PRESTO que l'intergiciel de communication, véritable Hub d'échanges entre les services de l'administration.

[7] Cf. http://europa.eu.int/idabc.

2. Défis architecturaux et axes de définition d'un protocole d'échange

Les solutions d'échange pour l'administration électronique sont souvent confrontées à des défis architecturaux ainsi qu'à des domaines réclamant une attention particulière (et qui sont, dans un premier temps bien souvent sous-estimés). Nous abordons ici la complexité et les difficultés, les options mises à disposition et l'approche préconisée ainsi que les axes de définition pour venir à bout de ces difficultés.

2.1 Défis architecturaux à résoudre

2.1.1 Multiplicité

Pour garantir la qualité des services de l'administration électronique et par voie de conséquence les échanges que cela suppose entre ces mêmes services, il est nécessaire de venir à bout de la diversité des systèmes et de faire appel, le cas échéant, à une plateforme commune consolidant et simplifiant la prise en charge de la totalité de ces services. Une telle plateforme est appelée dans la suite de ce livre-blanc une plateforme de services.

2.1.1.1 Accès/Intégration à une gamme de services élargie

Les initiatives d'administration électronique ne proposent, au départ, que quelques services, puis ne tardent généralement pas à se développer pour intégrer, au fil des ans, des services de plus en plus diversifiés. La pluralité des systèmes et leur nombre croissant ne cessent d'engendrer de nouvelles difficultés.

Les services dorsaux proposant ces services publics fonctionnent sur diverses plateformes et font appel à différentes technologies. La disparité des systèmes se fait de plus en plus sentir au fur et à mesure de l'expansion et de l'évolution des initiatives d'administration électronique. La conception et la mise en œuvre, dès l'origine, d'une infrastructure d'échange commune, capable de prendre efficacement en charge cette pluralité de services, constituent un réel défi qui dépasse de loin tous ceux habituellement rencontrés dans les systèmes commerciaux traditionnels.

Une fois en place et en production, une plateforme de services pour les besoins de l'administration électronique efficace doit pouvoir prendre en charge de nouveaux services ou types de documents/messages, et réagir aux modifications stratégiques en toute transparence, sans entraîner de modification du code, ni de perte d'exploitation risquant de nuire à d'autres services. Correctement gérée par la solution, la prise en compte de nouveaux services et/ou l'ajout de nouveaux messages ne doit plus relever de l'exception, mais de l'ordinaire.

Cette activité suppose pour être menée à bien de :

- Définir/Recenser les services (graduellement) accessibles à d'autres administrations ou agences gouvernementales,
- Mettre en œuvre un répertoire des services (graduellement) ainsi exposés,
- Définir et utiliser un langage commun de description de ces services,
- Et enfin, de partager les contrats de schémas et de services, en vue d'améliorer l'interopérabilité.

2.1.1.2 Pluralité des canaux d'accès

Ce n'est pas parce que les canaux de diffusion sont bien définis, dès l'aube du projet, pour les premiers services et applications et correctement pris en charge lors de la première mise en œuvre, qu'aucun autre besoin ne se fera sentir par la suite.

Une plateforme de services de confiance bien conçue et reliant/exposant les services avec cohérence, doit permettre l'intégration de nouveaux canaux, en une seule intervention très ponctuelle et sans interruption des échanges entre les services (et applications) individuels déjà en place.

2.1.2 Problèmes d'intégration

Tout comme les systèmes commerciaux les plus complexes, les plateformes de services pour les besoins de l'administration électronique doivent s'intégrer à un ensemble fini de systèmes généralement connu dès le départ, mais elles ont aussi d'autres impératifs à respecter. Elles doivent, en effet, fournir une infrastructure informatique qui soit capable de fédérer un ensemble plus vaste, voire inconnu, de services et d'applications afin de répondre aux besoins d'aujourd'hui et de demain. La réussite et l'adoption d'une plateforme de services dépendent de sa faculté d'intégration et de la continuité des échanges entre les services (et applications) existants.

L'intégration de systèmes hétérogènes (fonctionnant sur différentes plateformes et environnements techniques et équipés de logiciels variés) peut se faire de plusieurs façons. Nous présentons ci-après les options possibles. Le choix de l'option la plus adaptée dépendra, entre autres, des contraintes particulières à respecter, du type d'intégration voulu et du degré d'interopérabilité recherché.

2.1.2.1 Intégration native

L'intégration native entre deux systèmes n'est possible que s'ils sont compatibles au niveau voulu. Rappelons que la compatibilité peut exister au niveau de l'infrastructure et des réseaux, des modes d'accès aux données, des services et de leurs composants, de l'intégration des processus, des mécanismes de sécurité et d'identification et de la gestion des systèmes. Si deux systèmes parviennent déjà à communiquer, leur intégration ne demandera qu'un léger effort supplémentaire.

C'était généralement le cas lorsque deux systèmes fonctionnaient sur la même plateforme et utilisaient des logiciels, des modèles d'objets et des outils de développement compatibles. Les architectures orientées services permettent d'élargir le champ d'action en favorisant l'interopérabilité native d'une multitude de plateformes, à condition que ces dernières s'appuient sur des normes et standards du marché.

L'intégration native est l'« objectif ultime » que doivent viser, chaque fois que cela est possible, les systèmes d'administration électronique puisque l'effort à fournir pour une intégration réussie est alors bien moindre. Toutefois, l'architecture doit aussi tenir compte des cas où les contraintes qui pèsent sur les systèmes existants ou futurs, incapables de prendre en charge les interfaces SOA standards comme les services Web, empêcheraient toute intégration.

2.1.2.2 Installation d'adaptateurs sur la plateforme de services

Une autre approche tout aussi courante permet de décloisonner plusieurs systèmes, même lorsqu'ils résident sur des plateformes différentes. Elle consiste à installer des adaptateurs sur une plateforme de services afin d'accéder à des systèmes distants de type différent. La Figure 1 illustre cette approche globale où la plateforme dispose de quatre adaptateurs qui donnent accès à quatre services utilisant des technologies d'intégration différentes.

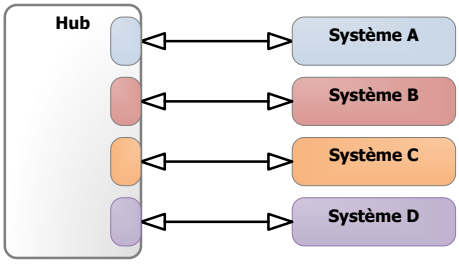

Figure 1 – Intégration de systèmes hétérogènes via l'installation d'adaptateurs sur la plateforme de services.

Cette solution présente l'avantage de laisser les systèmes distants inchangés et de confier à la plateforme le travail supplémentaire comme, par exemple, la liaison entre les protocoles réseau, la transformation des données vis-à-vis des messages échangés, le rapprochement sémantique et la gestion du flux des messages échangés.

Les protocoles en usage se décomposent généralement en un protocole métier spécifique associé à un protocole de transport spécifique. A titre d'illustration, de tels protocoles en usage au niveau de l'administration sont, par exemple :

- Pour la dématérialisation des pièces comptables, le protocole PES Hélios associé au protocole de transport Hélios ;
- Ou encore, sur le contrôle de légalité, le protocole métier Actes associé au protocole de transport Actes ;

D'autres protocoles comme CFT, FTP, Tedeco, PeSIT, etc. sont également en usage.

Cette approche amène aussi son lot d'inconvénients :

- Les divers protocoles natifs se propagent des systèmes distants jusqu'à la plateforme où résident les adaptateurs, ce qui surcharge (ou rend impossible) le trafic sur un réseau étendu et sur l'Internet ;
- L'intégration des nouveaux types de système impose une modification de la plateforme (ajout d'un nouvel adaptateur), ce qui peut entraîner l'interruption des échanges avec d'autres services (et applications) ;
- La capacité à monter en charge est limitée, surtout lorsque le nombre et la diversité des systèmes à intégrer ne cessent de croître au fil du temps ;

Pour toutes les raisons que nous venons d'invoquer, l'installation d'adaptateurs sur la plateforme de services ne répond pas à tous les cas de figure. Il convient de l'adopter en présence d'un petit nombre d'adaptateurs prenant en charge des standards bien établis et largement répandus.

2.1.2.3 Installation d'adaptateurs sur les systèmes distants

L'emploi d'« adaptateurs distants » est une variante de l'option précédente. Dans ce cas, la mise en correspondance des spécificités d'un système donné, comme la connectivité et les protocoles réseau, le format des données et la sémantique, se fait sur le système distant. Chaque système est responsable de sa propre intégration jusqu'au standard commun choisi.

Les services Web sont, sans conteste, le standard le plus largement répandu qu'il convient de recommander dans ce cas. Munis de l'adaptateur approprié, les systèmes distants peuvent s'intégrer de façon native à la plateforme de services. La quantité de travail à fournir en cas d'intégration personnalisée varie en fonction de l'écart qui sépare le système du standard, mais elle est généralement très proche de l'effort requis par l'installation d'adaptateurs sur la plateforme de services ; seul l'emplacement change.

Les adaptateurs peuvent faire partie du système distant d'origine ou être mis en œuvre séparément sur des plateformes proxy intermédiaires que l'on appelle mandataires émetteur et/ou récepteur. Cette deuxième solution s'avère très utile lorsqu'il est impossible de modifier les systèmes existants. La Figure 2 montre l'architecture de haut niveau où se situent les adaptateurs dans le système distant, sans pour autant en faire partie.

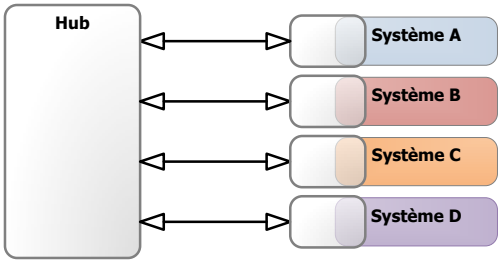

Figure 2 – Intégration de systèmes hétérogènes via l'installation d'adaptateurs sur les systèmes distants.

Cette approche présente les caractéristiques suivantes :

- Chaque système est intégré à un standard commun et l'intégration se fait en local sur le système cible, ce qui facilite l'opération ;
- Les communications vers les systèmes distants, qui s'établissent via un réseau étendu (WAN) dédié ou d'autres réseaux tels qu'Internet, sont standardisées ;
- Il est très facile d'ajouter de nouveaux services puisque la plateforme de services n'est pas concernée. Il est inutile d'installer de nouveaux adaptateurs sur celle-ci, et tous les services distants s'intègrent de la même façon ;
- Bien que l'effort d'intégration soit multiplié par le nombre de systèmes distants, il est possible de réutiliser plusieurs tâches puisque les différents adaptateurs donnant accès aux services individuels présentent généralement des similarités ;

2.1.2.4 Intégration générique ou personnalisée

Dans le cadre de l'intégration de systèmes distants à une plateforme de services, il convient de bien distinguer les fonctionnalités communes génériques (sécurité, fiabilité de la fourniture, stockage et transmission des messages, etc.) des tâches de personnalisation spécifiques et uniques à réaliser pour chaque système cible. Il est possible de reproduire la mise en œuvre de l'intégration générique d'un système à l'autre pour éviter tout effort redondant et inutile et de se concentrer sur les exigences requises par l'intégration personnalisée de chaque système cible.

La Figure 3 montre comment réutiliser certaines tâches d'intégration (repérées par un « X » sur la figure) sur plusieurs adaptateurs de façon à réduire les efforts de développement et à accélérer l'intégration.

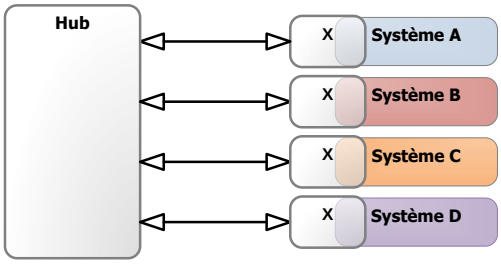

Figure 3 – Réutilisation des fonctions d'intégration génériques sur plusieurs adaptateurs distants.

2.1.2.5 Solution d'intégration commune et protocole d'échange

En toute logique, la prochaine étape consiste à définir une procédure commune pour mettre en œuvre les fonctionnalités d'intégration génériques dont chaque service distant peut bénéficier. Mieux vaut investir une bonne fois pour toutes dans la conception, le développement et les tests d'une solution d'intégration commune et s'en servir de base lors de l'intégration spécifique des services affiliés. Il s'agit de définir ici le protocole d'échange entre ces services (, applications) et/ou mandataires et de prendre en considération des différences de niveau, de besoins et de moyens : Administration centrale vs. Administration décentralisée.

Il est ainsi possible de réaliser de substantielles économies tout en assurant la qualité de la solution de base.

Ceci s'inscrit dans l'évolution progressive des réflexions de la plupart des États depuis une préférence imposée vers la promotion de l'interopérabilité et dans la volonté des instances publiques d'améliorer leurs systèmes informatiques hétérogènes, de « casser » les silos existants (qu'il s'agisse de données ou d'applications), afin de mieux coopérer entre services et entre administrations locales, nationales ou européennes et d'offrir ainsi un meilleur service aux citoyens.

La solution d'intégration commune proposée au travers du protocole d'échange a pour principale mission de garantir :

- L'asynchronisme, pour permettre non seulement le routage des messages entre plateformes de services et mandataires mais également la description des capacités du destinataire ultime ;
- L'optimisation du transport de documents au sens large (données fondées sur une grammaire XML ou non) ;
- La fiabilité de la transmission par laquelle tout message transmis arrive exactement une fois avec une gestion à la clé des acquittements de réception technique ;
- La sécurité comme l'intégrité du message transmis (via un cachet serveur par l'émetteur du message), ou encore la confidentialité du message transmis (chiffrement du message et des documents) ;

Ceci revient à définir une enveloppe technique de transport. La définition d'un standard d'échange pour les enveloppes des messages transmis entre deux systèmes d'information se doit d'être indépendante vis-à-vis :

1. Du protocole métier, en ce sens où les données transportées (contenu effectif et charge utile de l'enveloppe technique) sont nécessairement opaques ;
2. Des environnements, technologies et langages supportés et notamment de PHP, de Java/J2EE (nouvellement rebaptisé Java EE pour *Java Platform, Enterprise Edition*), du Framework .Net, etc. ;

Un tel standard se doit d'être facile et léger en termes d'intégration et de déploiement.

Enfin, ce standard se doit d'être ouvert au sens de la Loi pour la Confiance de l'Economie Numérique (LCEN) du 21 juin 2004 (2004-575) (http://www.legifrance.gouv.fr/WAspad/UnTexteDeJorf?numjo=ECOX0200175L) en son article 4) :

> « *Tout protocole de communication, d'interconnexion ou d'échange et tout format de données interopérable et dont les spécifications techniques sont publiques et sans restriction d'accès ni de mise en œuvre* »

Il lui faut aussi venir à bout d'autres difficultés, demandant souvent une expertise dont ne disposent pas nécessairement toutes les instances gouvernementales, surtout lorsque les services de l'administration électronique s'étendent aux établissements publics de moindre taille tels que les administrations locales. Dès lors qu'elle se fait en local sur le système cible (lorsque ceci s'avère possible), l'intégration personnalisée est bien plus facile et rapide à réaliser que s'il fallait procéder à une intégration complète jusqu'à la plateforme de services.

2.1.3 Souplesse et réactivité

En règle générale, les systèmes doivent être en mesure de réagir rapidement aux changements et de s'adapter aux nouvelles exigences. C'est d'autant plus vrai pour les plateformes de services pour les besoins de l'administration électronique où le rythme de croissance et les critères de volume, inconnus au départ, impliquent des changements beaucoup plus rapides. En outre, il s'avère bien plus nécessaire qu'ailleurs d'assurer la compatibilité amont avec les systèmes et versions déployés jusqu'alors.

2.1.3.1 Importance de la souplesse et de la réactivité

La plateforme de services pour les besoins de l'administration électronique doit pouvoir prendre en charge une vaste gamme de services (et d'applications) hétérogènes. Contrairement à beaucoup de systèmes commerciaux, où la gamme et la variété des services et des applications ainsi que les canaux (existants ou attendus) sont relativement bien définis, une plateforme de services pour les besoins de l'administration ou intergiciel de communication doit pouvoir anticiper les changements ou du moins y faire face dès qu'ils se présentent et, si possible, sans refonte ni interruption des échanges entre les services existants.

Plusieurs pays ont suivi un même scénario où les plateformes de services commencent par tenir un rôle de « pilote » et à assurer les interactions entre quelques services triés sur le volet en fonction de leur visibilité, de leur degré d'urgence et de leur aptitude à donner rapidement satisfaction. La réussite de la phase initiale aiguise les ambitions et les appétits. Il est alors question de prendre en charge davantage de services et d'applications et de compter sur la plateforme déjà en place pour les intégrer en toute transparence.

Il arrive que l'activation de nouveaux services et applications, de nouveaux fournisseurs d'authentification et de nouveaux canaux de diffusion donne lieu à une rénovation de la plateforme de services. Or, ces changements font partie d'une évolution normale et conventionnelle et, à ce titre, devraient être pris en charge par l'architecture, dès la conception de l'intergiciel de communication, des outils et des procédures. L'idéal serait de pouvoir modifier le système en place de la sorte sans aucune perte d'exploitation. Certains pays sont précurseurs dans ce domaine et ont prouvé qu'il pouvait en être ainsi.

2.1.3.2 Amélioration de la souplesse

Lors de l'architecture de la solution, il est essentiel de tenir compte du niveau de flexibilité que l'on cherche à atteindre. Pour faire face à des bouleversements majeurs (intégration de nouveaux services [et applications], ajout de nouveaux types de documents/messages, modification des méthodes de validation ou refonte des règles de routage des messages, par exemple) sans toucher au code, la solution doit être intégralement gérée par des paramètres de configuration et des données mises à jour au fur et à mesure des besoins.

Il est impératif d'adopter une architecture capable de prévoir et de prendre en charge la multiplicité à tous les niveaux (fournisseurs d'authentification, modèles d'identification et règles propres au service, par exemple). Compte tenu du fait que les exigences et les règles varient d'un service à l'autre, si ce n'est au début du moins ultérieurement, l'architecture doit prévoir le cloisonnement par service de façon à réduire l'ensemble des règles communes et des flux de messages sur lesquels les fournisseurs doivent s'entendre pour assurer le bon fonctionnement des échanges entre les systèmes. Pour enrichir l'offre de services, il convient d'éviter toute supputation quant aux types particuliers de connectivité, à la sécurité ou aux autres limites logistiques.

2.2 Axes de définition du protocole d'échange

La définition d'un protocole d'échange pour les besoins de l'administration électronique suppose de connecter la très grande variété des systèmes (et applications) à prendre en considération dans le contexte des échanges de l'administration électronique.

2.2.1 Construire les systèmes connectés

La construction de systèmes connectés ne constitue pas une nouveauté en soi. Durant les 25 dernières années, de nombreuses méthodes et technologies ont été utilisées pour résoudre les problèmes d'interopérabilité au sein d'une organisation et pour connecter cette organisation à ses partenaires. **L'interopérabilité se définit simplement comme la possibilité pour des systèmes, des composants et des services d'échanger des données et de l'information : en d'autres termes de « *se parler* ».**

Fort de cette définition simple de l'interopérabilité (nous ne rentrerons pas ici, car tel n'est pas le propos, dans une définition plus approfondie de l'interopérabilité), 2 systèmes techniques ou économiques sont **interopérables** s'ils peuvent collaborer sans se connaître intimement :

Utilisation uniquement d'échanges par messages, au moyen d'un transport partagé, suivant une « écriture » (schéma) partagée,
Sans préjuger lors de l'échange: du contexte, de l'interlocuteur, de l'implémentation du service demandé.

Diverses tentatives ont été menées dans le domaine de l'échange de documents.

Ainsi, **Electronic Documents Interchange (EDI)**, technologie mise en œuvre par certaines grandes entreprises (en tant qu'acheteur ou fournisseur), visait à remplacer des millions d'échanges de documents papier par des transactions électroniques.

Cependant, les promesses de l'EDI n'ont pas été complètement tenues dans la mesure où les solutions EDI résultantes ne sont pas extensibles, pas assez flexibles et ne peuvent donc pas être généralisées à toutes les communications inter entreprises : elles s'adressent avant tout à des scénarios B2B mettant en œuvre un faible volume et des messages volumineux. L'intégration des applications internes (EAI), la communication entre postes de travail ou le Pair à Pair (*Peer to Peer*), etc. constituent autant de périmètres non couverts.

Par ailleurs, les applications n'ont pas totalement éliminé le travail manuel nécessaire au traitement complet des documents. En fait, au lieu de réduire leurs ressources, certaines entreprises ont été contraintes de recruter des équipes pour traiter les documents reçus dans le cadre d'un processus manuel. Par ailleurs, les faibles améliorations de productivité et le coût additionnel induit par l'utilisation de réseaux spécialisés et de solutions EDI particulières ont empêché les entreprises de taille moyenne d'utiliser l'EDI pour connecter leurs entreprises. Cette incapacité à couvrir toute la chaîne de valeur, le manque de flexibilité et le coût de mise en œuvre et de fonctionnement ont empêché l'EDI de devenir la méthode de connexion entre tout type d'entreprise, sur tout type d'industrie et sur tout type d'interaction.

Sur la fondation *Extensible Markup Language* (XML), recommandation du W3C (*World Wide Web Consortium*) et langage universel de représentation de données, plusieurs modèles d'intégration ont émergé. XML permet à des systèmes hétérogènes de s'échanger des données indépendamment des systèmes d'exploitation, langages de programmation, base de données ou serveurs d'application utilisés et a donc fait rapidement son chemin comme standard de représentation de données pour l'entreprise.

electronic business XML (ebXML), projet commun entre UN/CEFACT (*United Nations Centre for Trade Facilitation and Electronic Business*) et de l'OASIS (*Organization for the Advancement of Structural Information Standards*) avait pour objectif de délivrer une série de spécifications destinées à faire évoluer l'EDI vers une infrastructure basée sur XML. Remplissant son cahier des charges initial, ebXML a effectivement délivré un socle suffisamment solide (exception faite de la sécurité) pour utiliser XML dans les scénarios B2B de type faible volume et messages volumineux.

Cantonné cependant au seul support des scénarios type EDI, ebXML échoue dans la réponse au besoin de généralisation des échanges par messages ; ceci constitue aujourd'hui la principale barrière à son adoption en dehors des marchés de niche de l'EDI. En effet, le protocole *ebXML Messaging Services* (ebMS) n'est pas conçu pour embrasser la diversité des scénarios à prendre en compte comme le souligne le Gartner dans son rapport « DIFFERENT GOALS MEAN EBXML IS NOT A RIVAL TO WEB SERVICES » (http://www.gartner.com/DisplayDocument?id=408059) :

> « *All organizations should emphasize development and implementation of Web services as a primary vehicle for IT transformation. Organizations and vendors that engage in business-to-business collaborative commerce should not view ebXML as a substitute for Web services, but as a work-in-progress with limited focus and international implementations that are currently limited. ».*

Toutes ces tentatives n'ont finalement intégré que la première dimension d'une autre approche pour atteindre l'interopérabilité, à savoir l'échange par messages, par un moyen de transport partagé, suivant une écriture (schéma) partagée. La seconde dimension qui ne préjuge en rien

de l'échange et du modèle de communication entre les systèmes a été occultée (partiellement ou totalement).

Ceci nous amène à imaginer de créer l'interopérabilité entre l'existant et le futur à travers une approche orientée services (*Service Oriented Architecture* ou SOA) en visant l'utilisation d'interfaces orientées messages, sans état. On rentre ainsi pleinement dans un « monde de protocoles ».

2.2.2 Architecture Orientée Services

Le principe fondateur d'une architecture orientée services réside dans l'idée que des systèmes, logiciels, périphériques mobiles et services « dialoguent » ensemble - même si ces derniers n'ont jamais été spécifiquement conçus en premier lieu les uns pour les autres et s'articule autour de 4 piliers.

- *Autonomie* **–** L'autonomie se définit comme la « liberté de gouverner par ses propres lois, indépendance, possibilité de disposer librement de soi ». Les services sont autonomes dans leur finalité et leur exécution. Un service ne doit pas avoir de dépendance forte vis-à-vis d'autres services, sinon on aboutit à un système fortement couplé qui est fragile et complexe.

  Un service est conçu, développé, déployé et géré comme isolé, indépendant des autres et interchangeable. Il fournit des fonctionnalités directement utilisables, indépendamment d'autres systèmes. Les services doivent maintenir toute l'information nécessaire pour opérer de manière isolée, de telle façon que si l'on a des services dont on dépend éventuellement, le service que l'on implémente n'en sera pas affecté : **il doit pouvoir fonctionner en dehors d'une connexion**.

- *Frontières explicites* **–** Les services étant autonomes, les frontières entre ceux-ci doivent être clairement définies. Les services peuvent être déployés sur des systèmes différents, dans des lieux différents, avec des technologies différentes, vis-à-vis de domaines de confiance différents. Il y a donc un prix à payer en termes de complexité et de performance pour traverser ces frontières.

  La gouvernance des frontières est reconnue explicitement en utilisant une technique d'échange explicite de messages entre les services. Les services sont exprimés à leur frontière et il n'y a qu'une seule façon d'accéder à l'information et/ou à la fonctionnalité au sein d'un service. Cette emphase mise sur les frontières de services permet de réduire la complexité de l'interaction entre les services et de formaliser les interactions entre les services indépendamment de leur implémentation (plateforme, langages, middleware, etc.). L'invocation d'une méthode est ici considérée comme une technique privée d'implémentation et non comme un moyen d'interaction entre services. Cette approche permet de réduire le nombre et la complexité des abstractions qui doivent être partagées par les services.

- *Partage de schémas et de contrats* **–** Les services sont autonomes, les frontières de communication sont explicites, les services masquent l'implémentation et communiquent par message avec l'extérieur. Les services doivent exposer leurs interfaces et les structures permettant l'échange d'informations à travers de schémas correctement construits, plutôt que sous la représentation d'une classe ou d'un type tel qu'imposé par un langage ou une plateforme.

  En adoptant des schémas fondés sur des standards ouverts, on peut réellement espérer – peut-être pour la première fois de l'histoire de l'informatique – créer des systèmes pleinement interopérables.

- *Politiques* – **Les services** négocient **en utilisant des « politiques ». C'est peut-être l'une des caractéristiques les plus fondamentales des architectures orientées services**. Les services doivent adhérer à la politique du tiers avec lequel ils désirent interagir pour interopérer. Si l'on offre un service fiable et transactionnel, on doit aussi supporter les protocoles sous-jacents, etc. La négociation protocolaire est alors effectuée dynamiquement et ceci permet aux concepteurs de services d'abstraire très largement les mécanismes d'encrage entre les systèmes et de se concentrer sur la sémantique fonctionnelle de leur service.

Les architectures orientées services conduisent à une réflexion générale d'urbanisation des systèmes d'information avec la disparition des applications monolithiques au profit d'une collection de services déployés indépendamment ; l'orchestration et la coordination (chorégraphie) des services étant l'un des points clé de la réussite de l'approche.

2.3 Approche préconisée

Les défis architecturaux abordés précédemment ont permis de laisser entrevoir la complexité, les difficultés et les options envisageables.

De ces dernières, nous retenons que le protocole d'échange pour les besoins de l'administration électronique est appelé à être mis en œuvre au niveau d'une plateforme de services et d'échanges de façon à fournir les enveloppes pour les messages transmis entre deux systèmes d'information d'administrations différentes. Ceci étant, cette plateforme est loin d'être la seule à mettre en œuvre ce protocole. Ainsi, l'ensemble des systèmes qui souhaitent directement communiquer et interagir avec cette dernière peuvent également disposer d'une implémentation logicielle de ce protocole.

Compte tenu de la multiplicité des systèmes (et applications) à prendre en considération dans le contexte des échanges de l'administration électronique, une approche conduisant à la spécification d'un protocole spécifique et monolitique suppose de disposer d'une implémentation non seulement – comme envisagé – au niveau d'une plateforme de services mais également sur l'ensemble des systèmes à connecter.

Cette approche entraîne, au-delà de son intrusivité potentielle, des coûts de mise à disposition extrêmement significatifs (multiples spécifications techniques d'implémentation, multiples développements, maintenance et évolutions, etc.).

A contrario, les spécifications standardisées des services Web de seconde génération permettent de répondre aux objectifs et **aux exigences d'échange** pour les besoins de **l'administration électronique**.

Proposées par les différents acteurs de l'industrie pour la diversité des systèmes à prendre en considération, les différentes implémentations d'ores et déjà disponibles ou à venir dans un futur proche, permettent de mettre à disposition à moindre coût un tel protocole d'échange.

Ceci étant, la plateforme de services doit offrir de toute évidence d'autres services que la simple implémentation d'un protocole d'échange et la définition seule du protocole ne saurait répondre aux multiples exigences d'intégration.

Il convient donc de rechercher une approche s'appuyant à la fois sur la fois sur la fondation des services Web abordée à la section précédente et intégrant le spectre complet des capacités d'intégration comprenant les fonctionnalités généralement communes aux produits regroupés sous la terminologie ESB (*Enterprise Service Bus*) comme :

- *Communication par courtage* **–** La fonctionnalité élémentaire consiste à envoyer des données entre processus situés sur le même système ou sur des systèmes différents. L'utilisation d'un logiciel intermédiaire entre l'émetteur et le destinataire offre une communication par courtier entre eux.
- *Routage intelligent et indirection d'adresses* **–** Ceci suppose un « référentiel » pour la résolution des adresses de service lors de l'exécution ainsi que la capacité de router les messages sur la base d'un jeu de critères prédéfinis.
- *Métadonnées des points terminaux* **–** Il s'agit de maintenir les métadonnées qui documentent les interfaces de service et les schémas de message.

Ainsi qu'au-delà de la validation, la transformation, le transcodage et la journalisation de messages, l'orchestration de processus métier.

Tous ces éléments ne sont pas du seul ressort d'un protocole aussi complet soit-il.

Il convient de définir dès lors non seulement un protocole d'échanges mais également d'envisager ici les interfaces externes et les services internes à offrir au niveau d'une plateforme de services.

3. PRotocole d'Échanges Standard et Ouvert

La spécification « *PRotocole d'Échanges Standard et Ouvert* »[8] 1.0 (alias PRESTO) publiée par la Direction Générale pour la Modernisation de l'État – Service pour le Développement de l'Administration Électronique (DGME SDAE) vise à fournir une couche générique d'échange de messages afin de pouvoir échanger potentiellement n'importe quels messages au sein de l'administration publique électronique dans le contexte de l'initiative ADELE entre les différents acteurs concernés par l'Ordonnance n° 2005-1516[9] sur les échanges électroniques.

Ce protocole concerne les échanges entre acteurs de l'administration électronique et n'a pas vocation à régir les échanges internes de chaque système d'information.

3.1 Périmètre et objectifs de conception

Parmi les projets concernés par ce protocole, figurent des programmes majeurs tels que Hélios pour la dématérialisation des pièces comptables ou encore Actes portant sur le contrôle de légalité.

Dans les deux cas, le développement des échanges électroniques entraîne une multiplication des flux de données et de documents de nature différentes : données basées sur une grammaire XML ou non.

Ces échanges posent également le problème de la sécurité des données. Si certains flux de données échangés entre systèmes d'information imposent une sécurité renforcée nécessitant l'authentification des documents transmis, d'autres ne présentent aucune mesure particulière.

La définition du protocole PRESTO s'est appuyée sur un état des lieux et une étude des trois protocoles existants en matière d'administration électronique :

1. FAST[10], solution d'infrastructure spécialisée de la Caisse des Dépôts et Consignation (CDC) dans l'envoi d'actes administratifs,
2. eLink[11], projet dans le cadre du programme IDABC de fourniture d'infrastructure générique européen abandonné fin 2006,
3. et ebMS 2.0[12], standard d'échange de l'OASIS en cours d'évolution vers une version 3.0 orientée Service Web.

La DGME SDAE a retenu les éléments techniques qui correspondaient le mieux aux attentes.

Ce protocole « idéal » a ensuite donné lieu à un appel à commentaires dont les retours ont permis d'affiner les besoins. Il est notamment ressorti la nécessité de prendre en compte les axes avancés dans le cadre de la section § 2.1.2.5 « SOLUTION D'INTEGRATION COMMUNE ET PROTOCOLE D'ECHANGE » en termes de garanties à offrir et d'indépendance mais également d'autres éléments développés comme :

- L'extensibilité (et l'évolutivité ou « scalabilité ») avec la possibilité de définir un canal PRESTO offrant différents niveaux de services ou encore la possibilité d'ajout de nouveaux services ;
- Ou encore le respect des normes et standards ouverts internationaux ;

[8] Cf. http://synergies.modernisation.gouv.fr/rubrique.php3?id_rubrique=165.

[9] Cf. http://www.legifrance.gouv.fr/WAspad/UnTexteDeJorf?numjo=ECOX0500286R.

[10] Cf. http://www.fast.caissedesdepots.fr/_fr/index.asp.

[11] Cf. http://ec.europa.eu/idabc/en/document/2322/5644.

[12] Cf. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebxml-msg.

3.2 Principes fondateurs

Les réflexions sur le protocole PRESTO se sont accordées sur la nécessité de prendre en compte les nouvelles spécifications standards des services Web. Des spécifications qui constituent les fondations du protocole PRESTO.

Il s'agit de reposer sur les promesses d'interopérabilité des services Web, l'une des traductions possibles de l'approche orientée services abordée dans la section § 2.2.2 « ARCHITECTURE ORIENTEE SERVICES ». Il s'agit de rationaliser les coûts d'investissement et d'exploitation, en tirant le meilleur parti de l'existant et des nouvelles plates-formes déployées. En effet, l'interopérabilité permet de favoriser le choix, la concurrence et l'innovation, de réduire les coûts et la dépendance vis-à-vis d'un fournisseur unique tout en favorisant un accès ouvert aux informations.

Cette orientation technologique reposant sur des standards ouverts internationaux trouve de multiples motivations. Elle :

- Constitue la clé du succès des initiatives d'administration électronique à la lumière notamment des prévisions du Gartner Group ;
- Promeut un accès ouvert à l'information et adresse les problèmes de compatibilité avec le passé (continuité de l'État dans le temps et sur tout le territoire) notamment au travers des possibilités de façades ;
- Entretient la bonne santé de l'écosystème informatique : choix, compétition et innovation ; les services Web étant aujourd'hui supportés par les principaux éditeurs de logiciels avec, à la clé, des solutions propriétaires ou Open Source ;
- Évite à un État d'être l'otage d'un fournisseur, de très nombreux éditeurs et autres acteurs de l'informatique proposant de telles solutions ou de l'expertise sur ces dernières ;
- Réduit les coûts, améliore l'efficacité, la flexibilité et la valeur ajoutée du système,

Dans ce contexte, revenons un instant sur ce que sont les fondements des services Web et leurs apports.

3.2.1 Capitaliser sur les fondements des services Web

Les services Web ont été conçus dès le début pour constituer un protocole de transmission de messages à même de répondre à tout scénario d'usage sans restriction aucune. Un grand soin a donc été pris pour s'assurer que leurs spécifications prennent en considération le plus large éventail de scénarios possibles et conditions de transmission de messages.

Ces mêmes services Web ont été conçus de façon à être modulaires et couplés de manière lâche avec un niveau élevé de qualité et d'uniformité technique à travers l'ensemble des spécifications. Ils représentent en cela une révolution des technologies distribuées, révolution fondée sur XML.

Pour ce faire, ils proposent une couche d'abstraction pour la découverte et l'invocation de services applicatifs. Ces services exposent des interfaces pour l'échange de documents. Ces interfaces peuvent être publiées et retrouvées dans un annuaire. Un client de se service doit simplement composer un message et l'envoyer à l'adresse du service.

Ils ne sont liés à aucune plateforme ou architecture mais sont fondés simplement sur des protocoles et des formats. Ce qui permet de se connecter avec virtuellement tous les systèmes existants et d'utiliser toute infrastructure existante.

Pour cela, **les services Web trouvent naturellement leurs racines dans les standards ouverts et notamment dans ceux de l'Internet et de l'*eBusiness* avec le W3C et l'OASIS**.

HTTP, SOAP, WSDL et UDDI constituent les fondements des services Web.

Ces qualités et le potentiel d'interopérabilité (qui en fait un modèle universel unique pour l'ensemble des interactions) font que les services Web reçoivent aujourd'hui une très large adoption de l'ensemble de l'industrie informatique et des clients.

En effet, la demande des clients vis-à-vis d'une plateforme de connexion commune conduit une large « palette » d'éditeurs - menée par Microsoft, IBM, BEA, SAP, Siebel, Sun, Iona, Oracle, etc. – à investir des milliards de dollars dans des produits et des solutions basés sur les services Web XML. Ce support universellement rencontré au niveau de chaque éditeur majeur couvrant des domaines allant des téléphones aux mainframes amène à la création d'un large éventail de produits qui supportent la pile des services Web XML. Ceci signifie que, virtuellement, pratiquement chaque nœud sur le réseau – chaque client, chaque serveur, et chaque service - peut envoyer et recevoir des messages services Web XML.

3.2.1.1 Protocole de communication standard

Exposé potentiellement sur Internet, le canal de communication de choix pour un service Web est tout naturellement le protocole *Hypertext Transfer Protocol* (HTTP, http://www.w3.org/Protocols), recommandation du W3C.

HTTP est un protocole de haut niveau (il se situe au-dessus d'un protocole transport comme TCP) qui enveloppe généralement une requête SOAP (Cf. section suivante) lorsqu'il sert dans le cadre d'un service Web. En réalité, n'importe quel protocole de niveau inférieur (couche transport) comme TCP ou UDP convient. Dès qu'un processus serveur peut recevoir et traiter des requêtes SOAP, tout autre protocole devient inutile. Alors pourquoi HTTP ? HTTP est un protocole très pratique, largement répandu puisqu'il est au cœur des navigateurs et du trafic Internet. L'avantage d'utiliser un serveur Web qui accepte déjà HTTP ou HTTPS sur les ports TCP 80 ou 443 réside dans le fait que toute l'infrastructure du réseau (les pare-feu, les routeurs, etc.) est déjà prête à travailler avec un trafic de ce type. Ainsi, les requêtes SOAP des services Web circulent très facilement dans cet environnement.

3.2.1.2 Format de représentation de données standard

L'échange de messages impose la capacité de représenter toute donnée. Cette représentation s'appuie sur XML (*Extensible Markup Language*, http://www.w3.org/XML), langage universel (*lingua franca*) de représentation de donnée et recommandation du W3C.

L'échange de messages suppose, au-delà de la capacité même de représenter les données afférentes, la définition de schémas et c'est tout naturellement le langage XML Schema (http://www.w3.org/XML/Schema) du W3C qui est utilisé.

L'échange de messages nécessite enfin de disposer d'une voie indépendante des plateformes pour permettre aux applications de communiquer les unes avec les autres au-dessus de l'Internet. C'est l'objet du protocole *Simple Object Access Protocol (SOAP) 1.2* [SOAP 1.2], recommandation du W3C.

Pour de plus amples informations, vous pouvez consulter le document « UNDERSTANDING SOAP » disponible à l'adresse Internet http://msdn2.microsoft.com/en-us/library/ms995800.aspx.

3.2.1.3 Langage de description standard WSDL

L'invocation d'un service suppose, à la base, la capacité de le décrire d'une façon compréhensible et exploitable de tous. C'est l'objet du langage *Web Service Description Language (WSDL) 1.1* [WSDL 1.1], langage formaté XML de description des possibilités d'un service Web. La description WSDL d'un service Web est obtenue dynamiquement.

Pour de plus amples informations, vous pouvez consulter le document « UNDERSTANDING WSDL » disponible à l'adresse Internet http://msdn2.microsoft.com/en-us/library/ms996486.aspx.

3.2.1.4 Mécanisme de découverte standard

Universal Description, Discovery and Integration (UDDI, http://www.uddi.org/about.html) crée une plateforme interopérable standard qui permet à des sociétés et à des applications de rechercher et d'utiliser rapidement, facilement, et dynamiquement des services Web au niveau de l'Internet.

UDDI constitue un effort inter-industries piloté par les principaux éditeurs de logiciels, mais également par des acteurs majeurs de l'industrie présents au sein du consortium OASIS. Microsoft a contribué à l'initialisation de ce projet.

L'usage d'un référentiel UDDI est recommandé par la DGME SDAE pour PRESTO. Pour les besoins et services de l'administration électronique, un tel référentiel est appelé à être déployé en France dans le cadre du projet de « *Répertoire d'Infrastructure Technique d'ADELE* »[13] (RITA) destiné à référencer les coordonnées techniques des destinataires de l'administration, en équivalence pages blanches.

3.2.2 Bénéficier des apports des standards de l'Architecture des Services Web

Au-delà des fondements précédents et pour atteindre ces objectifs (Cf. section § 3.1 « PERIMETRE ET OBJECTIFS DE CONCEPTION »), le protocole PRESTO repose, comme évoqué précédemment, sur les nouvelles spécifications standards des services Web.

Il s'agit plus exactement des spécifications/protocoles de la pile WS-* (* = STAR, qui signifie *Secured*, *Transacted*, *Asynchronous and Reliable* [ou Sécurisé, Transactionnel, Asynchrone et Robuste]), l'Architecture des Services Web (*Web Services Architecture* ou WSA) de seconde génération (interopérables) sécurisés, transactionnels, asynchrones et fiables. Les différentes spécifications/protocoles qui constituent cette pile sont, à ce jour, des recommandations du W3C ou des standards de l'OASIS (ou sur le point de le devenir).

Le livre blanc « INTRODUCTION A L'ARCHITECTURE DE SERVICES WEB ET SES SPECIFICATIONS WS-* » (http://www.microsoft.com/france/msdn/architects/intro-WS.mspx) donne une vue d'ensemble de la pile WS-* et des spécifications/protocoles qui la constitue.

Leur conception et la mise en œuvre associée ont été régies par différents principes fondamentaux particulièrement adaptés à la définition d'un protocole d'échange pour les besoins de l'administration électronique comme notamment l'orientation message et la composition.

L'orientation message qui conduit à une utilisation exclusive de messages pour communiquer entre services, tout en sachant que les messages ont souvent une durée de vie plus longue que leur temps de transmission. Les services Web supposent que SOAP est employé dans les couches basses de la pile des protocoles afin d'isoler le transfert des messages des détails de la couche transport. Ils ne s'appuient ni sur la syntaxe, ni sur la sémantique d'HTTP.

Les protocoles WS-* comme SOAP sont bâtis au niveau du message SOAP indépendamment du protocole de transport utilisé ; une dépendance stricte vis-à-vis d'HTTP pourrait, en effet, en limiter le cadre d'utilisation (pare-feu, passerelle, NAT, etc.).

En effet, les services Web peuvent répartir le traitement d'un message donné sur plusieurs nœuds du réseau (mandataires et plateformes de services et d'échanges), chacun contribuant à une fonctionnalité comme la vérification des accès, le routage en fonction du contenu du message ou une validation spécifique à une application. Ce modèle de traitement réparti implique qu'un message peut passer par deux ou plusieurs transports avant d'arriver à sa destination finale.

Pour cette raison, la plupart des premiers travaux sur les protocoles des services Web se sont concentrés sur la livraison fiable, sécurisée et de bout en bout de messages via des transports quelconques.

[13] Cf. https://www.ateliers.modernisation.gouv.fr/ministeres/domaines_d_expertise/architecture_techniq/public/projet_rita.

SOAP est défini indépendamment du mécanisme sous-jacent de transport des messages. Il autorise l'emploi de différents moyens de transport pour l'échange de messages, et permet à la fois des transferts et des traitements synchrones ou asynchrones.

Les protocoles WS-* sont construits au niveau du message SOAP et sont conçus pour être complètement indépendants de la couche de transport sous-jacente et donc du protocole de transport utilisé, la sélection du mécanisme approprié peut être retardée jusqu'au moment de l'exécution. Ainsi, les applications du service Web déterminent le transport approprié au moment de l'envoi du message. En outre, le transport sous-jacent peut changer pendant que le message passe de nœud en nœud indépendamment du protocole de transport utilisé.

Par ailleurs, plutôt que d'essayer de résoudre tous les problèmes possibles et imaginables, les spécifications individuelles de la pile WS-* essaient d'être courtes, concises et d'apporter une solution pertinente à un problème bien précis. Cette spécialisation des différentes spécifications permet à WS-* d'être utilisé comme un jeu de briques de standards pouvant être composées et assemblées orthogonalement pour composer un message SOAP avec le bon niveau de fonctionnalités (routage, sécurité, transactions, etc.). Il ne s'agit pas ici de définir un monolithe, mais bien des blocs pour construire les protocoles, blocs pouvant être utilisés dans pratiquement toutes les combinaisons.

Certaines spécifications peuvent être le cas échéant des compléments d'autres spécifications. Le livre blanc « SECURE, RELIABLE, TRANSACTED WEB SERVICES: ARCHITECTURE AND COMPOSITION » (http://msdn2.microsoft.com/en-us/library/ms996535.aspx) précise tous ces aspects.

Dans sa version initiale 1.0, le protocole PRESTO repose sur les spécifications/protocoles suivants de la pile WS-* :

- Simple Object Access Protocol (SOAP) 1.2 [SOAP 1.2]
- Web Service Description Language (WSDL) 1.1 [WSDL 1.1]
- WS-Addressing (proposition au W3C le 10 août 2004) [WS-AddressingAugust2004]
- SOAP Message Transfer Optimization Mechanism (MTOM) [MTOM]
- Web Services Reliable Messaging (WS-ReliableMessaging) 1.0 [WS-RM1.0]
- Web Services Security : SOAP Message Security 1.0 (WS-Security) [WS-Security]

La version 1.1 du protocole PRESTO supporte de façon additionnelle les spécifications/protocoles suivants :

- WS-Addressing 1.0 [WS-Addressing1.0] (en remplacement de [WS-AddressingAugust2004])
- Web Services Reliable Messaging (WS-ReliableMessaging) 1.1 [WS-RM1.1] (optionnel)

Nous développons ces éléments ci-après.

3.3 Format des messages PRESTO

Le format des messages est XML, ce langage étant rapidement devenu le format *de facto* pour les communications entre les applications (les consommateurs de services) et les services Web (les fournisseurs). XML fournit une structure qui donne un sens aux données. Ce n'empêche aucunement de véhiculer des données ne répondant pas à cette grammaire comme des données binaires.

Comme abordé précédemment, le W3C est l'organisation qui régit la spécification XML ; la recommandation officielle est actuellement la version 1.1. Consultez leur site à l'adresse http://www.w3.org. La spécification XML est disponible à l'adresse http://www.w3.org/XML.

L'engouement pour XML est dû à sa simplicité d'utilisation et au fait que ce format utilise du texte ordinaire. Ainsi, tout système, même le plus simple, dispose de la capacité de comprendre un message formaté en XML. La prise en charge de XML est désormais assurée par la quasi-

totalité des systèmes d'exploitation ; des bases de données acceptent même ce format comme type natif.

Ainsi, l'utilisation de XML pour fournir la structure de base des messages constitue un bon choix lorsqu'il s'agit d'assurer l'interopérabilité entre des systèmes et des applications, dans un environnement hétérogène.

3.3.1 Enveloppes, en-têtes et données

Le protocole PRESTO constitue une enveloppe technique. Le format « enveloppe » permet de placer les métadonnées à côté des données dans le message. Il inclut un en-tête qui contient la métadonnée, et un corps qui contient les données. SOAP constitue un bon exemple du format enveloppe : c'est un protocole fondé sur XML (adopté par la majorité des éditeurs de services Web) qui facilite la communication entre applications et empaquète les requêtes et les réponses avec les métadonnées associées. PRESTO utilise le format SOAP pour encapsuler des messages envoi Simple (*One Way*) émis à destination de services distants et/ou les requêtes/réponses échangées avec des services distants (Cf. section § 3.4 « ADRESSAGE DES MESSAGES PRESTO (OBLIGATOIRE) »). Cette approche permet à n'importe quelle plateforme exploitant n'importe quel système d'exploitation et s'appuyant sur n'importe quel langage de développement de supporter le protocole PRESTO.

Le protocole PRESTO repose sur la dernière version en date de la recommandation SOAP du W3C, en l'occurrence SOAP 1.2 [SOAP 1.2]. La version 1.2 de SOAP a atteint le statut de recommandation du W3C et représente, *de facto*, le consensus de l'industrie.

La Figure 4 décrit la structure d'un document avec enveloppe, cette dernière représentant la structure qui contient l'en-tête et le corps.

Figure 4 – Exemple d'un document au format enveloppe.

Il est possible d'imbriquer plusieurs structures de ce type. Ainsi, le corps d'un message SOAP peut à son tour contenir un autre message SOAP composé d'un en-tête et d'un corps. Cela implique plusieurs niveaux de métadonnées : la métadonnée associée au document (métier) et celle associée au message SOAP (technique, généralement en rapport avec la sécurité et la fiabilité).

Le protocole PRESTO définit un format enveloppe technique reposant sur SOAP et supporte, de facto, l'imbrication d'enveloppes, permettant ainsi au protocole PRESTO d'encapsuler, le cas échéant, d'autres protocoles comme le protocole IHE-XDS par exemple dans la Sphère Santé.

La Figure 5 montre deux enveloppes imbriquées, le corps de l'une incluant une autre enveloppe. Il est possible, si le besoin s'en faire sentir, d'ajouter d'autres niveaux d'imbrications.

Figure 5 – Exemple d'un document PRESTO contenant des enveloppes imbriquées.

La seconde enveloppe peut, par exemple, comporter des métadonnées qui répondent aux exigences fonctionnelles métier du protocole ainsi véhiculé.

Comme illustré dans la Figure 6, l'enveloppe technique PRESTO intègre dans son en-tête les services techniques attendus et s'appuie sur la composition des spécifications/protocoles WS-* afin de bénéficier du niveau d'interopérabilité attendue :

Figure 6 – Services techniques via l'enveloppe technique PRESTO.

L'envoi de tout type de message, quel qu'en soit les données, est donc supporté par le protocole PRESTO qui n'est pas intrusif à la différence d'un protocole comme FAST qui intègre des fonctions de validation de document.

Nous développons ces différents services ci-après ainsi que le support des pièces jointes aux messages PRESTO.

3.3.2 Pièce(s) jointe(s) aux messages PRESTO

Il est parfois nécessaire d'ajouter des pièces complémentaires à un document. Il peut s'agir de photographies, de la copie d'un permis de conduire ou d'un passeport, d'un document scanné ou de tous autres documents électroniques.

L'association de pièces jointes à un document XML (attachement) est possible techniquement via divers mécanismes et respecte plusieurs spécifications comme SOAP avec pièces jointes.

Le protocole PRESTO fait délibérément le choix d'un standard international, en l'occurrence la recommandation *SOAP Message Transfer Optimization Mechanism* (MTOM) [MTOM] du W3C comme illustré à la Figure 7.

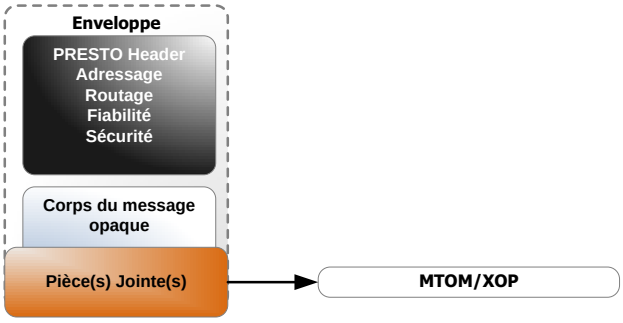

Figure 7 – Support des pièces jointes via les recommandations MTOM et XOP du W3C.

Cette approche offre l'avantage de ne considérer qu'un modèle de données : SOAP. La spécification SOAP a été conçue autour de la notion qu'un message SOAP est simplement une enveloppe SOAP basée sur XML : un message SOAP est un document XML qui consiste en une enveloppe obligatoire, un en-tête SOAP optionnel et un corps SOAP obligatoire. Ce document XML est référencé en tant que message dans toute la spécification.

Un message SOAP est spécifié comme un « *XML Information Set* », ou *InfoSet XML* (http://www.w3.org/TR/xml-infoset). La perspective de revisiter SOAP pour la prise en compte d'un nouveau modèle de données est intimidante compte tenu du nombre de problématiques à

aborder de nouveau. De plus, en abandonnant un modèle basé sur XML, le traitement des messages SOAP perdrait le bénéfice des spécifications modernes de XML comme les schémas XML (*XML Schema*, http://www.w3.org/TR/xmlschema-0), les requêtes XPath (*XML Path Language*, http://www.w3.org/TR/xpath) ou *XML Query* (http://www.w3.org/TR/xquery) et les transformations XSL (*Extensible Stylesheet Language*, http://www.w3.org/TR/xsl). Ceci offre également un support aisé d'intermédiaires SOAP, mandataires PRESTO de routage par exemples ou plateformes de services et d'échanges au sens large tel qu'envisagés au chapitre § 2 « DEFIS ARCHITECTURAUX ET AXES DE DEFINITION D'UN PROTOCOLE D'ECHANGE ».

MTOM est une spécification qui décrit comment optimiser la transmission des messages SOAP (en particulier l'enveloppe) ; elle fait référence à la spécification *XML-binary Optimized Packaging* (XOP) [XOP], recommandation du W3C, pour décrire comment l'optimisation est mise en œuvre. Le service Web qui reçoit le message SOAP est responsable de reconstruire le message d'origine, en décodant le contenu binaire optimisé en une représentation Base64, bien que cette étape ne soit pas obligatoire.

Le format du pack d'optimisation binaire XOP utilise MIME pour permettre l'inclusion de données binaires brutes dans un document XML. À l'emplacement des données encodées Base64 dans l'InfoSet XML, un élément XOP spécifique établit un lien vers un contenu binaire optimisé XOP placé dans le message MIME. Le paquet résultant (après sérialisation de l'InfoSet) est nommé paquet XOP ; il contient le document XML (document XOP) et le contenu binaire dans un format MIME Multipart/Related. MTOM spécifie comment lier ce format à SOAP.

Avec cette approche, en plus du codage en mode texte de l'InfoSet, un codage de l'InfoSet permettant à des données binaires opaques d'être placées au milieu d'un balisage traditionnel en mode texte est pris en charge. Autrement dit, MTOM, avec XOP, décrit un mécanisme pour l'optimisation de la transmission et/ou du format sur le câble d'un message SOAP en ré-encodant de façon sélective des portions du message tout en présentant à l'application SOAP un *InfoSet* XML.

L'utilisation du standard MTOM dans le cadre de PRESTO amène les trois bénéfices clés vis-à-vis des tentatives précédentes :

1. *Composition* – Le potentiel de composition de MTOM est excellent. En effet, le résultat final d'un transfert MTOM est une enveloppe SOAP ; de fait, l'ensemble des protocoles services Web de plus haut niveau fonctionne tel que conçu.

   En particulier, MTOM compose avec la sécurité (via le standard WS-Security de l'OASIS), ce qui se traduit par la sécurisation des données binaires au niveau message SOAP/PRESTO. Les recommandations XML-DSIG et XML-ENC du W3C sont directement utilisables comme décrit dans la section § 3.7 « SECURISATION DES MESSAGES PRESTO ».

2. *Efficacité* – Avec MTOM, les valeurs de caractère binaire sont transmises sur le réseau comme attachement MIME et sont référencés dans le corps du message SOAP.

   Bien que le transfert de données MIME Multipart ne soit pas aussi efficace que transférer des données brutes comme avec DIME, il s'agit tout de même d'un transfert de données sans inflation de la taille.

   En composition avec WS-Security, MTOM ne diminue pas le temps de traitement au niveau client ou serveur pour la sécurisation du message. Ceci réside dans le fait que WS-Security requiert que les données soit encodée en Base64 afin d'appliquer les algorithmes de canonisation et de normalisation et donc de générer les valeurs chiffrées.

3. *Modèle de mise en œuvre/programmation simplifié* – Chaque éditeur ayant décidé de la supporter, la recommandation MTOM du W3C a bien évidemment sa propre feuille de route.

   Pour ce qui est de Microsoft, le support de MTOM est amené par la version 3.0 du Microsoft Framework.NET.

La spécification SwA (*SOAP with Attachments*, http://www.w3.org/TR/SOAP-attachments), une liaison pour qu'un message SOAP soit véhiculé au sein d'un message MIME (*Multipurpose Internal Mail Extensions*) a été écartée dans la mesure où cette dernière n'offre pas la possibilité d'être composée avec la spécification WS-Security pour les services de technique de sécurité.

3.4 Adressage des messages PRESTO (obligatoire)

Pour dialoguer avec un service Web, il est vital que chaque partie comprenne le protocole d'échange de messages, en l'occurrence ici le protocole PRESTO.

La Figure 8 illustre l'utilisation de la spécification WS-Addressing dans le cadre de PRESTO (proposition au W3C le 10 août 2004) [WS-AddressingAugust2004] ou WS-Addressing 1.0 [WS-Addressing1.0] selon la version du protocole PRESTO) pour la transmission des messages sur les réseaux indépendamment du transport employé, implique que chaque partie (application ou mandataire) comprend comment l'autre traite le message et quel emplacement sert à la livraison des messages.

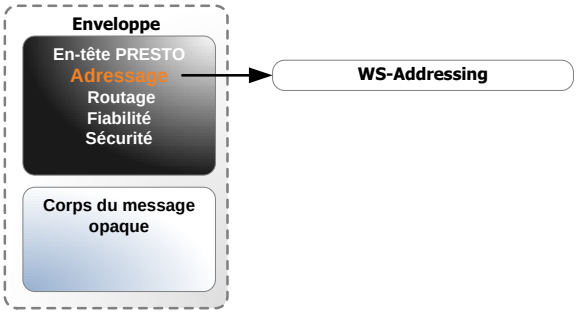

Figure 8 – Adressage des messages PRESTO via la recommandation WS-Addressing du W3C.

Le standard WS-Addressing du W3C définit une façon standard pour s'adresser aux points terminaux d'un service Web en utilisant une référence de point terminal.

Il définit aussi comment convoyer l'information (routage du message) et comment invoquer un service en utilisant les en-têtes d'information de messages. Cela signifie que la représentation d'un flux de message synchrone ou asynchrone s'effectue de façon standardisée et indépendante du transport dans un en-tête SOAP. Ce ne serait pas le cas si l'on utilisait certains en-têtes de protocoles de transport pour véhiculer des informations relatives à des requêtes de services Web. Par exemple, l'en-tête SOAP-ACTION dans HTTP décrit l'opération SOAP au niveau du transport ; actuellement, il est utilisé dans la majorité des services Web reposant sur HTTP. Le fait de déplacer cette information dans l'en-tête SOAP (elle fait ainsi partie du message) permet de transmettre le message SOAP via n'importe quel protocole réseau et pas seulement HTTP.

Actuellement, cette base permet la prise en charge par le protocole PRESTO des principaux motifs d'échange de messages (*Message Exchange Patterns* ou MEP) suivants :

- Envoi simple (*One Way*) de message – Il s'agit d'un envoi sans attente de réponse avec un traitement ultérieur ;
- Échange de message Requête/Réponse – L'émetteur reste en attente d'une réponse ;

Les évolutions suivantes seront prises en compte dans une prochaine version du protocole PRESTO :

- Routage de message via un intermédiaire SOAP ;
- Échange de message avec un tiers (via un autre protocole) ;

3.5 Routage des messages PRESTO (non couvert actuellement)

Le routage des messages n'est pas couvert dans les premières versions des spécifications PRESTO.

Figure 9 – Service de routage via l'enveloppe technique PRESTO à définir.

Un des principaux objectifs du service de soumission consiste à router des documents vers la destination adéquate. Ce routage peut s'effectuer de concert avec un référentiel UDDI comme RITA de la DGME SDAE.

3.6 Fiabilité des échanges de messages PRESTO (obligatoire)

L'envoi d'un message à un destinataire (application ou mandataire récepteur) n'implique en lui-même aucune garantie que le message sera livré au destinataire final. L'expéditeur (application ou mandataire émetteur) n'a aucun moyen de savoir si le destinataire a bien reçu le message et s'il est nécessaire de renvoyer le message. De plus, certains messages doivent être transmis dans un ordre précis. Dans son expression la plus simple, SOAP ne définit aucun mécanisme pour livrer les messages dans l'ordre.

L'utilisation obligatoire du standard *Web Services Reliable Messaging* (WS-ReliableMessaging) 1.0 [WS-RM1.0] ou 1.1 [WS-RM1.1] de l'OASIS dans le cadre de PRESTO comme illustré par la Figure 10 assure la transmission des messages avec une garantie de fiabilité, notamment en cas d'incident sur le réseau. Un mécanisme d'acquittement de messages est mis en place entre l'émetteur et le récepteur pour garantir la livraison des messages.

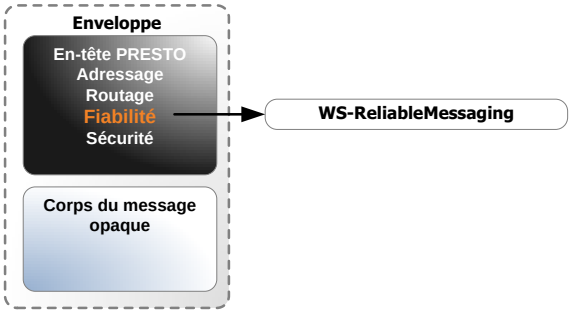

Figure 10 – Fiabilité des messages PRESTO via le standard WS-ReliableMessaging de l'OASIS.

Prenons l'exemple d'une application qui livre un message à un nœud destination. Le nœud destination accuse réception en renvoyant un message à l'application source. Le chemin de routage est fourni par les différents messages utilisés dans WS-ReliableMessaging, comme *CreateSequence*, *LastMessage*, *SequenceAcknowledgement* et *TerminateSequence*.

Au niveau du protocole PRESTO, la *Source fiable de message*, lorsqu'elle reçoit un message de l'application émettrice ou d'un mandataire émetteur, émet un message *CreateSequence* destiné à la *Destination fiable de message*. La destination, un mandataire récepteur par exemple, crée un identificateur de séquence et renvoie un message *CreateSequenceResponse*. Puis un ou plusieurs messages sont échangés en incrémentant le

numéro du message à chaque fois. Le dernier message est signalé par le jeton *LastMessage*. La destination répond avec un message *SequenceAcknowledgement* qui contient des informations sur les messages reçus. Si certains manquent, la source les retransmet en demandant à la destination de les acquitter. Lorsque tous les messages ont été reçus par la destination, la *Source fiable de message* envoie un message *TerminateSequence*.

La Figure 11 décrit le mécanisme de dialogue mis en œuvre par le protocole WS-ReliableMessaging. Dans cet exemple, le message 2 échoue. Le protocole le détecte et demande sa retransmission par la source.

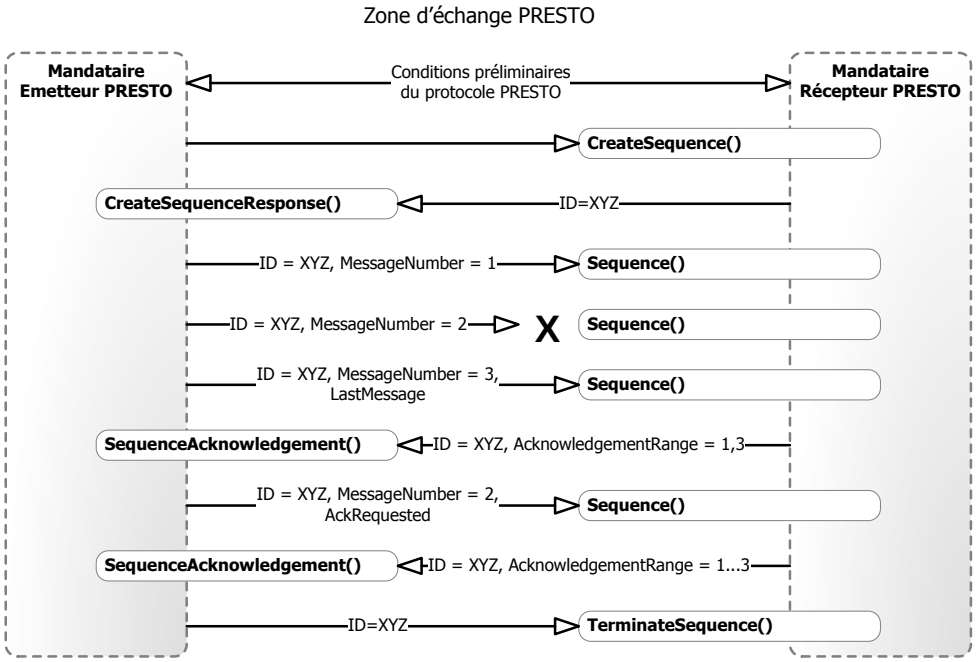

Figure 11 – Mécanisme utilisé pour assurer la fiabilité des messages avec WS-ReliableMessaging.

En outre, le standard WS-ReliableMessaging respecte l'ordre entre les messages ; ainsi, le récepteur reçoit les messages dans l'ordre dans lequel ils ont été envoyés. Le protocole PRESTO utilise l'option *ExactlyOnce* pour assurer la livraison des messages.

3.7 Sécurisation des messages PRESTO (optionnelle)

SOAP n'est pas conçu pour sécuriser les données qu'il transporte. Comme abordé précédemment, il s'agit d'une spécification de la structure élémentaire d'un message (enveloppe, en-tête et corps).

Cette absence de mécanisme de sécurité au début du développement des services Web, et en particulier dans SOAP, oblige à utiliser d'autres mécanismes pour assurer la sécurité. Ces mécanismes dépendent de l'environnement, par exemple de la couche transport si l'on utilise des protocoles comme SSL, TLS ou IPSec. Malheureusement, la sécurité au niveau de la couche transport est souvent inadaptée.

La sécurité appliquée au niveau de la couche transport assure la confidentialité du contenu entre l'expéditeur du message et le destinataire. En dehors de cette liaison établie entre l'expéditeur et le destinataire, le message n'est plus protégé ; il circule généralement en clair. Dès lors, la sécurité du message est dépendante de l'architecture réseau et des plateformes sous-jacentes et de la confiance que l'on peut leur donner si tant est que ces dernières soient connues.

Pour cette raison, aucune garantie ne peut être donnée que le message soit resté privé et confidentiel lorsqu'il arrive au niveau du destinataire ultime. Si le message est passé par des nœuds intermédiaires, sa sécurité peut être compromise lors du passage d'un réseau à un autre.

Au sein des système d'information des administrations, pour des raisons de performance et d'efficacité, le chiffrement au niveau transport est souvent assuré par des matériels dédiés, situés à la frontière avec Internet. La session n'est donc plus sécurisée à l'intérieur du Systèmes d'Information, en-deçà de ces points terminaux. Même lorsqu'une authentification mutuelle a eu lieu, des informations sur l'origine du demandeur peuvent être perdues lorsque le système chiffre le message.

De façon à ne rien présupposer en termes d'intermédiaires SOAP (de routage par exemple) et offrir des garanties de sécurité pour le message à transmettre entre l'émetteur initial et le destinataire ultime, le protocole PRESTO s'appuie sur le standard largement adopté de l'OASIS WS-Security (« *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS Standard 200401, March 2004* », http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf) comme illustré par la Figure 12 suivante :

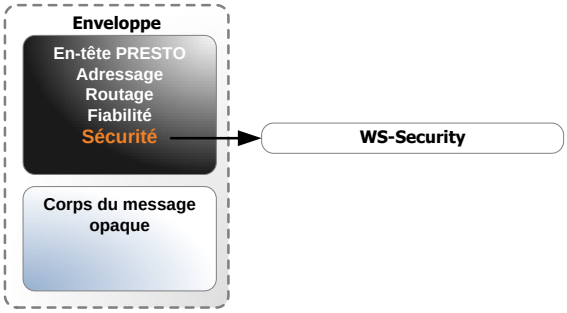

Figure 12 – Sécurité optionnelle des messages PRESTO via le standard WS-Security de l'OASIS.

Cette approche permet d'offrir, de façon optionnelle, une sécurité au niveau du message et de ses données tend à dépasser les problèmes rencontrés avec la sécurité au niveau de la couche transport.

WS-Security définit, en effet, un ensemble d'en-têtes SOAP qui servent à garantir le respect de la confidentialité et de l'intégrité des messages envoyés d'un émetteur initial jusqu'au destinataire ultime, quels que soient les réseaux traversés, le nombre de nœuds intermédiaires et les plates-formes rencontrées sur le trajet.

3.7.1 Authentification et autorisations

Le protocole PRESTO permet de véhiculer dans l'en-tête SOAP de l'enveloppe technique un ou plusieurs jetons de sécurité qui identifie/authentifie voir confère des autorisations pour l'émetteur initial (ou son délégué) vis-vis (d'un intermédiaire ou) du destinataire ultime pour le document qu'il reçoit. L'émetteur initial doit s'être authentifié auprès de son domaine de sécurité avant de soumettre le document, et, en supposant, le cas échéant, qu'il existe un cadre de confiance (direct, indirect, par courtier, par délégation, etc.), entre les domaines de sécurité de l'émetteur et du destinataire si ces derniers diffèrent. Selon le type de jeton, une signature numérique appliquée au jeton garantit qu'aucune falsification ne peut s'immiscer lors du transfert et des contrôles d'intégrité sont possibles sur le message PRESTO.

Après vérification et décision de validation du ou des jetons de sécurité, le destinataire peut s'appuyer sur les mécanismes de contrôle d'accès en place. Un contrôle pourra s'assurer, par exemple, que le type d'authentification et/ou le niveau d'autorisations de l'émetteur initial dans son domaine de sécurité correspond au niveau minimal requis par le destinataire cible. Dans le cas d'une réponse négative, un message d'erreur SOAP est renvoyé pour indiquer que le document ne peut être soumis au service destinataire.

3.7.2 Intégrité

Il se peut qu'un message PRESTO ne soit ni sensible ni confidentiel. Par conséquent, il ne nécessite pas de chiffrement. Toutefois, l'émetteur initial peut vouloir s'assurer que le contenu du message ne sera pas modifié durant le transfert du message jusqu'au destinataire ultime. Le mécanisme mis en œuvre pour obtenir ce résultat se nomme cachet serveur qui s'apparente à une signature numérique.

Apposer un cachet Serveur revient à utiliser un algorithme de hachage comme SHA-1 (ou SHA-2 aujourd'hui compte tenu des récents progrès réalisés en cryptanalyse des condensés) afin d'obtenir une valeur numérique de longueur fixe et de taille réduite, nommée hachage ou condensât de message. Ce hachage est unique et caractérise de façon univoque le document source. Toute modification insérée dans le document produit une modification dans le hachage.

Toutefois, lorsqu'on applique une fonction de hachage à des documents XML, un problème apparaît : deux documents XML, identiques d'un point de vue syntaxique, peuvent donner des résultats différents après hachage en raison d'espaces ou de retours chariot en plus. Or, les espaces ou les retours chariot affectent le formatage du document, pas sa syntaxe ni ses données. Par conséquent, il est nécessaire d'exécuter un algorithme de canonisation sur le document XML avant d'exécuter la fonction de hachage afin d'éliminer les espaces selon des règles prédéfinies.

Le W3C a publié une spécification nommée Canonical XML (*C14N Canonicalization*, http://www.w3.org/TR/xml-c14n) qui décrit les règles de la canonisation des documents en vue de leur signature. C'est une recommandation en version 1.0.

Dans un message SOAP qui utilise WS-Security, il est possible d'appliquer un cachet Serveur sur des parties quelconques du message, y compris les éventuelles pièces jointes. WS-Security s'appuie sur la spécification XML Signature qui décrit les conventions à respecter pour signer numériquement un message SOAP.

La spécification XML Signature (*XML Signature Syntax and Processing* ou XML-DSIG, http://www.w3.org/TR/xmldsig-core) est une recommandation du W3C.

3.7.3 Confidentialité

Dans un message PRESTO, le chiffrement est possible à n'importe quel niveau. Ainsi, il est possible de chiffrer tout ou partie des données y compris les éventuelles pièces du message. Même les en-têtes SOAP peuvent être chiffrés. WS-Security s'appuie sur la spécification *XML Encryption* qui décrit les conventions à respecter pour chiffrer un message SOAP.

La spécification *XML Encryption* (*XML Encryption Syntax and Processing* ou XML-ENC, http://www.w3.org/TR/xmlenc-core) est une recommandation du W3C.

3.8 Relation avec les standards internationatiaux

Comme illustré précédemment dans la description des différents services techniques du protocole et du support des pièces jointes aux messages PRESTO, PRESTO repose sur des standards internationaux du W3C et de l'OASIS, conformément à ces objectifs initiaux de conception.

Si ces derniers sont aujourd'hui largement adoptés, il n'en demeure pas moins que des problématiques d'interopérabilité peuvent néanmoins exister. L'amélioration du potentiel d'interopérabilité passe par la définition de profils d'interopérabilité.

Dans la pratique, le protocole s'apparente à un profil d'interopérabilité des recommandations et standards suscitées. Sa définition capitalise et s'inspire en grande partie du profil WS-RAMP[14] (*Reliable Asynchronous Messaging Profile*) 1.0 développé par IBM pour l'industrie automobile américaine.

Dans le principe, deux parties souhaitant communiquer via des services Web doivent simplement se mettre d'accord sur la syntaxe et la sémantique des messages échangés. Une fois ce contrat défini, elles sont libres de choisir les langages de programmation, les systèmes, les machines virtuelles ou les bases de données.

[14] Cf. http://www.ibm.com/developerworks/webservices/library/specification/ws-ramp.

La définition de profils d'interopérabilité pour les services Web fait partie des missions de l'organisation WS-I. Afin d'accélérer le développement et le déploiement de l'interopérabilité des services Web et donc, de garantir que les différents standards supportant les services Web soient réellement implémentés de manière interopérable entre les différents éditeurs du marché, Microsoft et 45 autres entreprises ont fondé l'organisation WS-I. Les membres du « board » sont Accenture, BEA Systems, Fujitsu, Hewlett-Packard, IBM, Intel, Microsoft, Oracle, et SAP. L'organisation comprend d'autres membres participants parmi les utilisateurs de services Web comme Daimler Chrysler, Ford Motor, Qwest, Reuters, et United Airlines. Sun a rejoint récemment cette organisation qui comprend aujourd'hui plus de 130 membres.

La première mission de WS-I est de garantir l'interopérabilité par l'usage des services Web qui promettent d'intégrer des applications développées et maintenues par différentes organisations sur différentes technologies. Les services Web développés avec les outils existants n'interopèrent pas toujours de manière satisfaisante : des méthodes de services Web ne peuvent pas être exposées, des méthodes peuvent être exposées mais pas consommées ou encore des types de données ne peuvent être sérialisés. Ces incompatibilités sont dues à l'utilisation de différents types de message SOAP ou à des différences d'interprétation des interfaces WSDL. Les éditeurs d'outils ont travaillé à corriger ces problèmes, mais la rapide évolution de SOAP et des standards complémentaires ainsi que la gestion des différentes versions d'un même standard rendent ce travail difficile. De plus, les éditeurs doivent s'entendre non plus sur une seule, mais sur plusieurs technologies ne serait-ce qu'au niveau du fondement des services Web comme nous l'avons vu (HTTP, XML Schema, SOAP, WSDL, UDDI).

WS-I peut simplifier cette tâche en définissant des profils qui correspondent un groupe de spécifications compatibles à un certain niveau de version adressant un ensemble cohérent de fonctionnalités et de recommandations d'utilisation à observer. Un profil est un ensemble de spécifications, accompagné de clarifications, interprétations et restrictions de celles-ci de façon à promouvoir l'interopérabilité.

Dans le même temps, WS-I crée des protocoles et outils de test (http://www.ws-i.org/implementation.aspx) afin de vérifier le respect effectif de ces profils par les différents produits et solutions du marché. Les éditeurs proposent conjointement des modèles de mise en œuvre qui se conforment aux recommandations précédentes (http://www.ws-i.org/implementation.aspx). En ce sens, WS-I est une sorte d'intégrateur de standards qui s'appuie et qui utilise les spécifications des différents organismes comme le W3C, OASIS ou IETF.

Ainsi, vis-à-vis du fondement des services Web, WS-I a déjà publié le profil élémentaire « WS-I BASIC PROFILE 1.1 », (http://www.ws-i.org/Profiles/BasicProfile-1.1.html) qui prend en considération SOAP 1.1, XML Schema 1.0, WSDL 1.1, et UDDI 2.0.

Ce profil constitue aujourd'hui **LA référence en matière d'interopérabilité des services Web**. Ce dernier est en cours de normalisation au niveau de l'ISO/IEC.

En raison de la prolifération de plates-formes et de technologies différentes dans l'administration publique, il est essentiel que les services Web soient capables de fonctionner dans un environnement hétérogène. En l'état de la définition du profil, PRESTO offre la meilleure adhérence à ce profil, à l'exception de la version 1.2 de SOAP qui n'est pas prise en compte.

Au fur et à mesure que les spécifications de l'architecture des services Web deviennent des standards, l'organisation WS-I s'intéresse à la définition de profil élémentaire correspondant. C'est le cas aujourd'hui pour le standard de l'OASIS WS-Security 2004 dont les implémentations deviennent omniprésentes.

De nouveaux profils sont en cours de définition au niveau du WS-I. Il s'agit plus particulièrement des profils suivants :

- « WS-I BASIC PROFILE 1.2 »[15] - Cette version du profil élémentaire WS-I introduit la prise en compte des recommandations WS-Addressing et MTOM du W3C ;
- « WS-I BASIC PROFILE 2.0 » - Cette future du profil élémentaire WS-I sera basée sur les spécifications suivantes : SOAP 1.2, XML InfoSet, WSDL 1.1 and XML Schema 2001 pour la description des services, WS-Addressing, MTOM, XOP, UDDI v2 & v3 pour la découverte et la publication ;
- « WS-I RELIABLE SECURE PROFILE (RSP) 1.0 »[16] - Ce profil prend en compte les spécifications WS-ReliableMessaging et WS-SecureConversation. Les travaux sont en cours au niveau de l'organisation WS-I ;

Aujourd'hui, PRESTO est un profil de services Web défini par la DGME SDAE sur la base de standards W3C et OASIS. Demain, PRESTO sera une simple restriction de ces profils voire même s'effacera purement et simplement au profit de ces derniers.

3.9 Evolutions programmées

Au-delà des caractéristiques courantes du protocole ou profil PRESTO telles qu'illustrées ci-avant, une prochaine version de PRESTO utilisera les stratégies pour définir les métadonnées associées aux points terminaux en relation. Elle sera aussi capable de demander et de consommer dynamiquement ces stratégies selon les spécifications des services Web WS-Policy [WS-Policy], WS-RMPolicy [WS-RMPolicy], WS-SecurityPolicy [WS-SecurityPolicy] et WS-MetadataExchange [WS-MetadataExchange].

3.10 Documentation et implémentations disponibles

Dans sa version courante, le protocole PRESTO est décrit avec :

- La référence technique PRESTO [PRESTO-Ref] qui fournit un profil normatif pour l'ensemble des spécifications des services Web sur lequel reposent les exemples du Kit de démarrage PRESTO,
- Le guide PRESTO [PRESTO-Guide] qui fournit une description générale du modèle d'échange de messages PRESTO, en tant qu'enveloppe technique.

Ces documents sont accessibles à l'adresse Internet https://www.ateliers.modernisation.gouv.fr/ministeres/projets_adele/a131_b_protocole/public/presto/folder_contents. Pour obtenir des informations complémentaires sur le protocole PRESTO, veuillez vous reporter aux documents cités dans la section § « REFERENCES ».

En termes de support des spécifications WS-* sur lesquelles repose PRESTO, chaque éditeur a bien évidemment sa propre feuille de route. Le colloque du 17 octobre 2006 organisé par la DGME dans ses locaux a permis de présenter quelques démonstrateurs issus de différents acteurs :

Au niveau des produits et technologies Microsoft, le support des spécifications WS-* est amené par la version 3.0 du Microsoft .NET Framework.

Le Microsoft .NET Framework 3.0 (auparavant connu sous le nom de WinFX) est le nouvel modèle de programmation de code managé pour la plateforme Windows. Il combine la puissance du .NET Framework 2.0 avec de nouvelles techniques pour le développement d'applications qui communiquent très facilement au-delà des frontières technologiques, et qui sont capables de prendre en charge un large éventail de processus métiers.

[15] Cf. http://www.ws-i.org/Profiles/BasicProfile-1.2.html.

[16] Cf. http://www.ws-i.org/deliverables/workinggroup.aspx?wg=reliablesecure.

Parmi ces innovations se trouve notamment Windows Communication Foundation (WCF). Auparavant nommé « Indigo », WCF constitue le framework unifié que propose Microsoft pour le développement d'applications distribuées sécurisées, fiables, transactionnelles et interopérables. Parmi beaucoup d'autres choses, WCF offre une implémentations complète des spécifications WS-* sur lesquelles repose le protocole PRESTO. Le modèle de programmation orienté service correspondant s'appuie sur le Microsoft .NET Framework.

Pour des informations complémentaires sur la technologie WCF, veuillez consulter les ressources suivantes :

- Le site MSDN dédié sur http://msdn2.microsoft.com/en-us/netframework/default.aspx
- Le site communautaire sur http://www.netfx3.com.

Disponible par défaut avec Windows Vista, le package redistribuable Microsoft .NET Framework 3.0 à destination des plateformes Windows XP SP2, et Windows Server 2003 SP1 et ultérieurs est disponible par téléchargement gratuit à l'adresse Internet suivante : http://www.microsoft.com/downloads/details.aspx?FamilyId=10CC340B-F857-4A14-83F5-25634C3BF043.

Sur cette base, Microsoft France a récemment publié sous le contrat de licence de logiciel libre CeCILL-B[17] le Kit de démarrage PRESTO pour le Microsoft Framework .NET 3.0 qui propose une implémentation évolutive du profil PRESTO à destination d'applications ou de mandataires émetteur ou récepteur.

Ce kit correspond pour l'essentiel au démonstrateur développé pour les tests d'interopérabilité du profil PRESTO et présenté lors du colloque PRESTO du 13 octobre 2006. Le démonstrateur réalisé et démontré a notamment permis de couvrir l'ensemble des tests d'interopérabilité avec 100% de réussite vis-à-vis du prototype réalisé par la société Sun Microsystems en Java sur la technologie GlassFish (http://glassfish.dev.java.net).

La description de ce prototype est présentée à l'adresse Internet http://blogs.sun.com/alexismp/resource/ProtoSunMicrosystems_fr.html.

Par ailleurs, afin d'illustrer une intégration du protocole PRESTO dans un outil de tous les jours pour un utilisateur final, le Kit de démarrage PRESTO pour le Microsoft Framework .NET 3.0 comprend un add-in d'extension pour la suite bureautique Microsoft Office 2007. Cet add-in, écrit également en C# et .NET 3.0, permet d'envoyer avec le protocole PRESTO le document courant en cours d'édition à un destinataire ou mandataire récepteur PRESTO. Le document ainsi, qu'il s'agisse d'une feuille de calcul, d'un document de texte ou d'une présentation. Ce document peut être au format ouvert Open XML, standard international de l'Ecma.

Ce kit de démarrage est disponible par téléchargement à l'adresse Internet suivante : http://www.microsoft.com/downloads/details.aspx?FamilyID=B8D92FE4-2D93-491E-B30C-E58CD9808AA0.

Ce kit est également disponible sur AdmiSource, la plateforme collaborative proposée à l'ensemble des administrations Françaises pour leurs développements de logiciels libres, à l'adresse internet : http://admisource.gouv.fr/projects/presto.

[17] Cf. http://www.cecill.info/licences/Licence_CeCILL-B_V1-fr.txt.

4. Plateforme de services et d'échanges

4.1 Vision et éléments d'architecture

Comme introduit dans le cadre de la section § 2.1.2 « PROBLEMES D'INTEGRATION », une infrastructure commune à de nombreux fournisseurs de services de l'administration électronique résout la plupart des problèmes posés par l'intégration des services. Cette infrastructure se nomme plateforme de services et d'échanges pour l'administration électronique, ou plus simplement Hub, qui, en complément d'un protocole d'échange standard et ouvert doit fournir les catégories suivantes de services :

- Services de sécurité et d'identité (fédérée) ;
- Services de messagerie ;
- Services de routage et d'intégration ;

Nous avons développé notre vision et les éléments d'architecture afférents dans un précédent livre-blanc intitulé « PROTOCOLE D'ECHANGE POUR LES BESOINS DE L'ADMINISTRATION ELECTRONIQUE ». Ce livre-blanc peut être téléchargé à l'adresse Internet http://download.microsoft.com/download/6/5/B/65B32DAE-D3F7-4CE0-ADF3-8F060B822BA6/Protocole_echange_pour_besoins_administration_electronique.pdf.

Nous renvoyons le lectorat à ce livre-blanc pour une présentation de l'approche que nous préconisons et des grands principes d'architecture qui régissent une telle approche ainsi que les éléments de mise en œuvre associés.

Alors que la mise en place d'un tel Hub accroît le nombre de services l'utilisant, rien n'impose la mise en place d'un Hub unique pour l'ensemble des services de l'administration électronique.

Plusieurs Hubs peuvent coexister, mettant en œuvre à des degrés divers un sous-ensemble de la gamme complète des fonctionnalités et coopérant les uns avec les autres dans diverses topologies, selon un modèle fédéré. L'objectif est de fournir une flexibilité s'accommodant de diverses contraintes, de topologies et d'échelles d'implémentation différentes, tout en respectant le même plan directeur que pour des nœuds individuels. Les scénarios les plus simples correspondent à des cas bien délimités, capables d'évoluer vers des solutions plus complexes par ajout de capacité ou de fonctionnalités, sans remettre en cause l'existant.

Le Hub implémente un ensemble de services mis à la disposition de différents types d'entités externes. Le Hub lui-même se présente comme une entité lorsqu'il appelle d'autres entités (systèmes et/ou services) externes. Le précédent livre-blanc explore les topologies possibles différentes autorisées et/ou à prendre en considération :

- *Hub unique, plateforme commune à tous les services* – Il implémente des fonctions essentielles comme la gestion des identités, la sécurité et le prétraitement (comme le routage) des messages, c'est-à-dire des documents, fournis par les clients. Les fournisseurs de services de l'administration publique sont déchargés de ces tâches et peuvent se concentrer sur le traitement de fond des documents reçus.
- *Hub vu comme une plateforme d'intégration des systèmes « back-end »* – Une version réduite du Hub générique, ne servant qu'à l'intégration des systèmes « back-end » de traitement, se révèle très efficace pour interconnecter les systèmes du fournisseur de service.

  Le Hub d'intégration peut exploiter les fonctions de messagerie, de routage et d'intégration pour les messages reçus, fonctions fournies par un autre Hub ou par d'autres systèmes. Il effectue la validation nécessaire et passe les messages aux systèmes cibles. En plus des fonctions génériques de messagerie et d'intégration, le Hub peut héberger des fonctions spécifiques de transformation et d'intégration liées au service public concerné ou aux systèmes « back-end ».

Le Hub d'intégration peut faire appel à un Hub principal pour valider et authentifier les messages entrants. Il peut aussi effectuer ses propres contrôles pour authentifier et autoriser les requêtes. Ainsi, un dialogue peut s'établir entre le Hub d'intégration et des services externes, comme le Hub principal, notamment dans le cas où les messages proviennent de Hubs qui ne sont pas approuvés.

- *Hubs de services en relation homologue à homologue* – Dans de nombreux cas, il n'est guère efficace de faire passer tout le trafic via un Hub principal, notamment si ce Hub n'apporte pas de valeur ajoutée intéressante (par exemple, il n'effectue aucune tâche de validation, d'autorisation ou de suivi). Une communication homologue à homologue devient alors préférable. Elle est utilisée seule ou en complément de communications avec le Hub principal, pour des types particuliers de messages. Ce genre de communication permet, par exemple, de renvoyer une réponse directement au client, même si la requête provient du Hub principal. Elle sert aussi dans certaines phases de conversation incluant des messages multiples, après la phase initiale de dialogue.

  Des instances du Hub générique facilitent des communications efficaces d'homologue à homologue, et effectuent des tâches standards qui, sinon, devraient être mises en œuvre dans les services eux-mêmes. Selon le type des interactions et des fonctions requises, il est envisageable de déployer des sous-ensembles différents de la fonctionnalité d'un Hub générique.

- *Hiérarchie de Hubs de services* – Il est possible d'envisager des topologies plus complexes, où des Hubs de services à plusieurs niveaux coopèrent dans un modèle distribué et fédéré. Certaines des fonctions du Hub générique (comme les décisions de routage, d'orchestration ou d'autorisation à différents niveaux de granularité) peuvent être réparties entre des Hubs de services, en prenant en compte des considérations comme les performances, la disponibilité, le propriétaire des données, les procédures de maintenance et de mise à jour, etc. Au lieu de rechercher une solution unique « à tout faire », plusieurs solutions collaborent pour faire face à des contraintes et à des exigences variées.

D'autres scénarios sont encore possibles. La fonctionnalité générique du Hub peut être répartie entre plusieurs instances du Hub, afin de pouvoir faire face à de nombreux cas particuliers.

Ces aspects sont également couverts par le livre-blanc précédent.

De façon à donner la dimension opérationnelle qui est la sienne au protocole PRESTO, nous souhaitons nous concentrer sur l'illustration d'un point de vue technique, avec les produits et technologies Microsoft, de la mise en œuvre d'un tel Hub, corollaire du protocole PRESTO. C'est l'objet de la section suivante.

4.2 Mise en œuvre avec Microsoft BizTalk Server 2006 R2

La plateforme Microsoft BizTalk Server 2006 R2 offre la fondation nécessaire pour mettre en œuvre la notion de plateformes de services et d'échanges ou tout simplement de Hub comme envisagé ci-dessus.

Un tel Hub peut être dissocié en deux Hubs comme suit :

1. *Hub de messages* - Ce Hub est responsable de l'échange fiable et sécurisé de documents électroniques entre plusieurs points terminaux. Dans le contexte de l'administration électronique, il simplifie les transactions C2G (grand public - administration) et B2G (entreprises - administration), et offre une plateforme pour l'échange de messages électroniques entre les acteurs de l'administration électronique ;
2. *Hub d'intégration* - Le Hub d'intégration représente la contrepartie du Hub de messages dans chaque entité de l'Administration électronique et facilite l'intégration avec les systèmes « back-end » de l'établissement. Ce Hub transforme et garantit la livraison des documents XML entre les entités cohérentes, comme un ministère, un service de collectivité, un partenaire social, etc. et des Hubs de message, tout en maintenant un historique des transactions et en signalant les incidents. Il présente les caractéristiques suivantes :

- Livraison garantie des documents XML entre l'entité et le Hub de messages, incluant un magasin de stockage interne et de transfert, une répétition automatique de la soumission et une livraison unique. Elle s'appuie sur la technologie RM (messagerie fiable) du BizTalk Framework ;
- Intégration souple et sans écrire de code avec les systèmes « back-end » grâce à BizTalk Server. Le Hub d'intégration prend en compte de nombreux protocoles, formats de données et adaptateurs d'application, ainsi que des outils graphiques pour la définition de messages, la transformation et l'orchestration ;
- Transformation des messages en un format compréhensible par les systèmes « back-end » ;
- Notification des incidents de communication avec les systèmes « back-end » ;
- Historique des transactions et archivage des documents soumis ;

4.2.1 Vue d'ensemble

Vis-à-vis de l'implémentation des deux Hubs précédents, BizTalk Server 2006 R2 (http://www.microsoft.com/biztalk/default.mspx) propose deux éléments fondateurs principaux :

1. Un moyen de spécifier les types de processus métier,
2. Ainsi que des mécanismes pour communiquer/consommer les messages de ces applications sur des transports les plus divers au titre desquels se trouve un support des nouvelles spécifications standards des services Web (Cf. section § 3.2.2 « BENEFICIER DES APPORTS DES STANDARDS DE L'ARCHITECTURE DES SERVICES WEB »).

Ces éléments sont adressés par les composants suivants du moteur de BizTalk Server 2006 R2 comme illustrée par la Figure 13 :

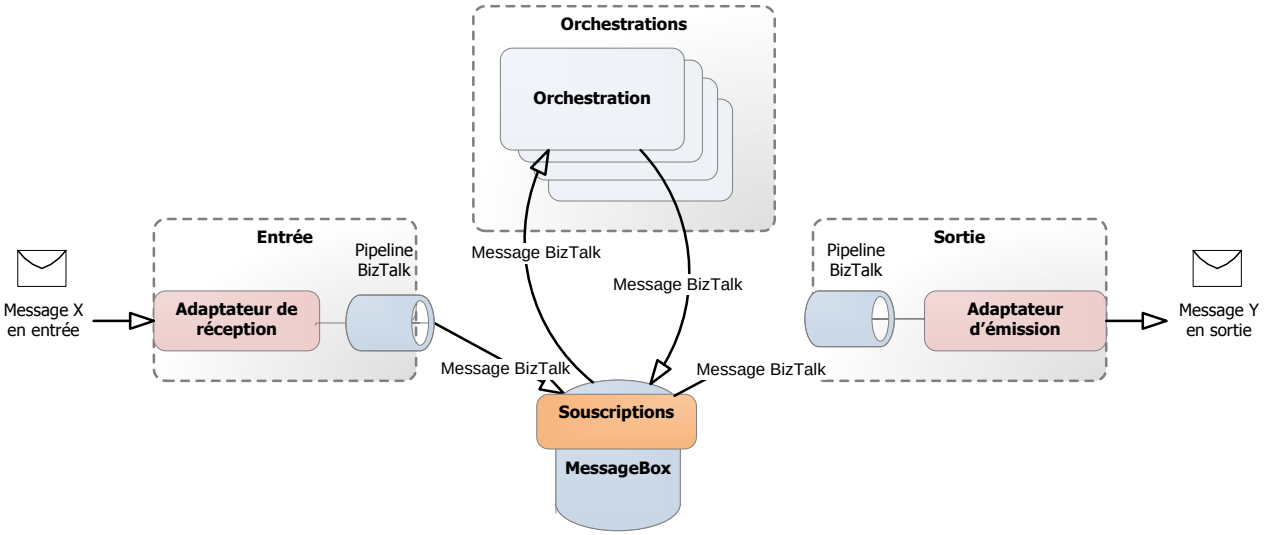

Figure 13 – Composantes de BizTalk Server 2006 R2

R2 est construit sur la fondation de BizTalk Server 2006 et n'introduit pas de changements au niveau de l'architecture. Nous retrouvons donc les mêmes principes de fonctionnement.

Un message envoyé vers BizTalk Server peut être véhiculé par différents protocoles. La prise en charge par BizTalk Server d'un nouveau message suppose sa réception par un *adaptateur de réception* dont le type varie suivant le mécanisme de communication mis en œuvre :

- Nouveau fichier disponible dans un répertoire,
- Nouveau message dans une file d'attente Microsoft Message Queuing (MSMQ),
- Appel par Web service,

- Etc.

Différents adaptateurs correspondent à différents mécanismes de communication pour le bus de messages. Ainsi, la disponibilité d'un tel adaptateur vis-à-vis de tel ou tel protocole et notamment dans notre contexte plus particulièrement du protocole PRESTO permettent à BizTalk Server d'agir en qualité de mandataire d'émission PRESTO. La Figure 14 illustre cette caractéristique :

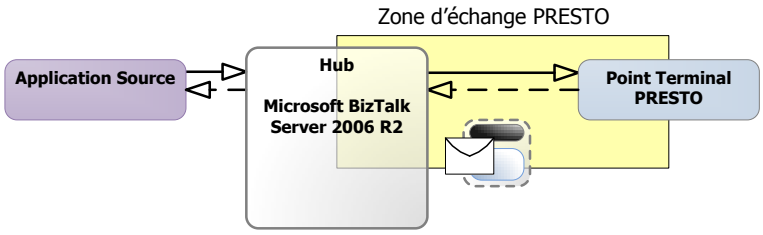

Figure 14 – Mandataire d'émission PRESTO

De façon similaire, la disponibilité d'un adaptateur pour PRESTO en réception permet à BizTalk Server d'agir en qualité de mandataire de réception PRESTO. La Figure 15 illustre cette caractéristique :

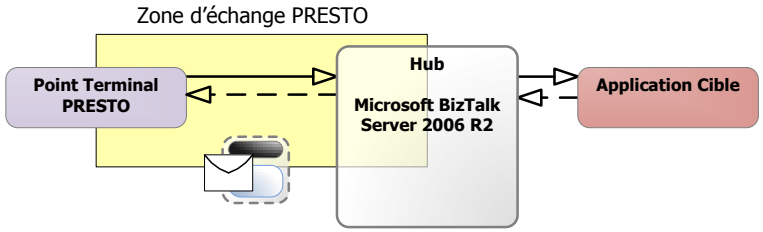

Figure 15 – Mandataire de réception PRESTO

Les adaptateurs PRESTO proposés dans le cadre du Kit de démarrage Adaptateurs PRESTO pour Microsoft BizTalk Server 2006 R2 sont décrits dans la section éponyme § 4.2.4 « KIT DE DEMARRAGE ADAPTATEURS POUR BIZTALK SERVER 2006 R2 ».

BizTalk Server supporte aussi bien les flux asynchrones que synchrones; le choix de l'un des deux modes s'effectue lors de la conception d'un port logique de réception (*receive port*) des messages. Au sein de l'environnement BizTalk Server, les messages ne sont vus que dans un seul format : XML. Cependant, les messages susceptibles d'être reçus dans l'adaptateur sont de formats variables (XML, mais aussi fichiers plats, fichiers binaires, documents IDOC, EDIFACT, etc.). Pour être converti au format XML, le message reçu doit être traité via un *pipeline de réception*. Ce pipeline peut contenir divers composants qui réalisent des actions telles que la validation d'un cachet d'émission (signature numérique de l'émetteur du message), le déchiffrement du message, la conversion du format natif du message vers un format de document XML, etc.

Le message est ensuite acheminé vers une base de données située au cours du dispositif et appelée *MessageBox*, laquelle repose sur Microsoft SQL Server. Une fois créé dans cette base, le message est susceptible d'être acheminé vers une orchestration ou d'être directement routé sur d'autres protocoles, dans d'autres formats (*content-based delivering*).

Un processus métier est mis en œuvre sous la forme d'une ou plusieurs *orchestrations*, chacune d'entre elles correspondant à du code exécutable. Cependant, ces orchestrations ne sont pas créées en écrivant du code, via un langage tel que C# mais définies graphiquement via un groupe défini de formes permettant d'exprimer les conditions, boucles et autres comportements du processus métier. Les processus métier peuvent également utiliser le moteur des règles de gestion qui offre un moyen plus simple et plus aisé pour définir les règles d'un processus métier.

Chaque orchestration crée des *souscriptions* afin d'indiquer le type de messages qu'elle s'attend à recevoir. Lorsqu'un message répondant à ces critères arrive dans la *MessageBox*, il est expédié vers son orchestration cible, laquelle effectue ensuite les actions requises par le processus métier. Le résultat de ce traitement a typiquement pour effet la production d'un autre message, par le processus métier, lequel est sauvegardé dans la *MessageBox*.

Ce nouveau message (ou le message d'origine, s'il n'y pas eu d'orchestration) est à son tour traité par un pipeline d'émission, qui peut le convertir du format interne XML utilisé par BizTalk Server vers le format requis par sa destination, qui aura pour tâche d'éventuellement convertir le message XML au format souhaité en sortie, d'apposer un cachet serveur (signature numérique), de le chiffrer, etc. Le message est ensuite envoyé via un *adaptateur d'émission*, qui utilise un mécanisme approprié pour communiquer le message à l'application cible.

Enfin un *adaptateur d'émission* véhicule le message sortant du pipeline vers sa cible, en mettant en œuvre le protocole de communication souhaité (SMTP, SOAP, PRESTO, etc.).

4.2.2 Intégration des applications au sein du bus de message

Une bonne intégration passe par un échange efficace des messages entre les applications et services. Même si ce bus se veut avant tout un bus d'échange de messages XML sur le protocole HTTP, étant donné la diversité des modes de communication qui existent, une grande variété de protocoles et de formats de message doit être supportée.

C'est ce qu'offre le moteur de BizTalk Server ; une portion significative de celui-ci étant dédié à établir cette communication. Le moteur ne fonctionne en interne qu'avec des documents XML. Quel que soit le format dans lequel un message arrive, il est systématiquement converti en document XML après sa réception. De même, si le récepteur d'un document ne peut accepter un format XML, le moteur le convertit dans le format de message attendu par l'application cible.

4.2.2.1 Réception et émission de messages

Compte tenu de la diversité évoquée ci-dessus, le moteur de BizTalk Server repose sur une gamme d'adaptateurs pour réaliser les opérations de réception et d'émission de messages. Un adaptateur est l'implémentation d'un mécanisme de communication tel qu'un protocole particulier.

L'ensemble des adaptateurs disponibles pour BizTalk Server 2006 R2 ont été développés en se fondant sur l'Adapter Framework. Dans le contexte de ce livre-blanc, on peut noter avec R2 un support étendu d'EDI avec la présence d'un adaptateur Base EDI qui permet l'envoi et la réception de messages via X12 and EDIFACT.

Plus de 300 adaptateurs tierce partie sont, par ailleurs d'ores et déjà disponibles. Une liste complète est disponible à l'adresse Internet http://www.microsoft.com/biztalk/evaluation/adapter/default.mspx. Ils ont donc tous un socle commun, ce qui facilite grandement leur utilisation par BizTalk Server, et la production de nouveaux adaptateurs.

Dans le même temps, des accélérateurs pour RosettaNet et d'autres standards de l'industrie comme par exemple HIPPA (*Health Insurance Portability and Accountability Act*) et HL7 pour la Sphère de la Santé sont disponibles. Les accélérateurs sont énumérés sur la page « BIZTALK ACCELERATORS PRODUCT OVERVIEW » (http://www.microsoft.com/technet/prodtechnol/biztalk/2006/evaluate/overview/accelerators.mspx).

Par ailleurs, au-delà des fonctionnalités offertes par les adaptateurs destinés à une connectivité IBM, BizTalk Server peut interagir directement avec la plateforme de communication *Host Integration Server 2006* (http://www.microsoft.com/hiserver) afin de disposer d'une connectivité SNA étendue (intégration dans des transactions IBM CICS ou IMS, accès à des données DB2, fichier plat, etc. sur des mainframes MVS ou des systèmes départementaux AS/400).

Quel que soit l'adaptateur utilisé pour recevoir des messages, ces derniers doivent être traités avant d'être accessibles par une orchestration. De même, les messages de sortie d'une orchestration ont souvent besoin d'être traités avant d'être envoyés par un adaptateur.

Avec cette capacité de communication étendue avec les protocoles les plus divers, BizTalk Server offre en natif la logique nécessaire pour fonctionner comme une passerelle lorsqu'une application tierce est la destination cible d'un message PRESTO, c'est-à-dire comme une application de courtage pour permettre l'intégration d'application et l'intégration d'entreprise à entreprise. La Figure 16 illustre la possibilité de :

1. Traduire les messages PRESTO de façon sophistiquée en messages tierce partie,
2. Router les messages vers le réseau tierce partie, tout en prenant en compte les caractéristiques du protocole tiers.

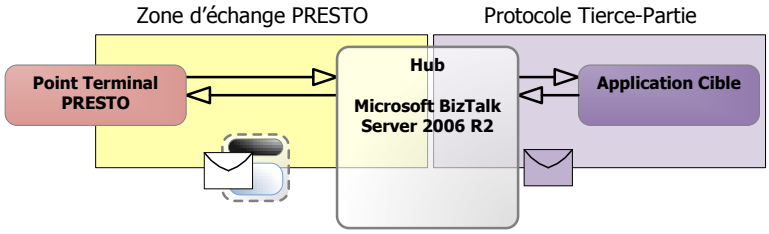

Figure 16 – Passerelle/Application de courtage vers une application cible

Bien évidemment, qui peut le plus peut le moins et la logique de BizTalk Server peut servir d'intermédiaire de routage de messages PRESTO avec ou sans transformation sur la base ou non de l'exécution de processus métier comme illustré par la Figure 17.

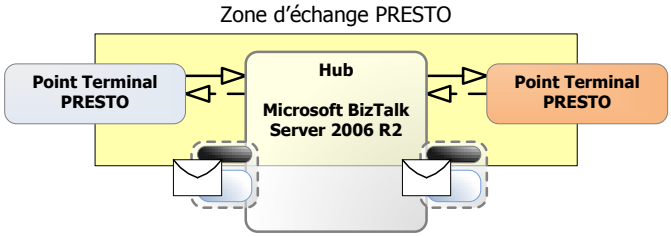

Figure 17 – Intermédiaire de routage de messages PRESTO

4.2.2.2 Traitement des messages avec les pipelines

Les services et applications sous jacentes à un processus métier communiquent en échangeant des types de documents variés. Une application BizTalk Server qui exécute un processus métier doit être capable de traiter correctement les messages qui contiennent ces documents. Ce traitement peut impliquer de multiples étapes et est réalisé par un pipeline message. Les messages entrants sont traités par un pipeline de réception, tandis que les messages sortants le sont par un pipeline d'émission.

Ainsi, même si de plus en plus d'applications sont capables d'interpréter les messages XML, le moteur de BizTalk Server ne travaillant en interne qu'avec des messages XML, il doit offrir un moyen de convertir les autres formats en XML et vice versa. D'autres services peuvent être également requis, tel que l'authentification de l'émetteur du message. Afin de gérer ces tâches d'une façon modulaire et composable, un pipeline est construit à partir d'un certain nombre de phases, chacune d'entre elle contenant un ou plusieurs composants .NET ou COM. Chaque composant gère une partie spécifique du traitement du message. Le moteur de BizTalk Server offre plusieurs composants standardisés qui adressent la plupart des cas courants. Ceci étant, si nécessaire, des composants personnalisés peuvent être créé à la fois pour le pipeline d'émission et pour le pipeline de réception.

La Figure 18 montre les phases et composants d'un pipeline de réception.

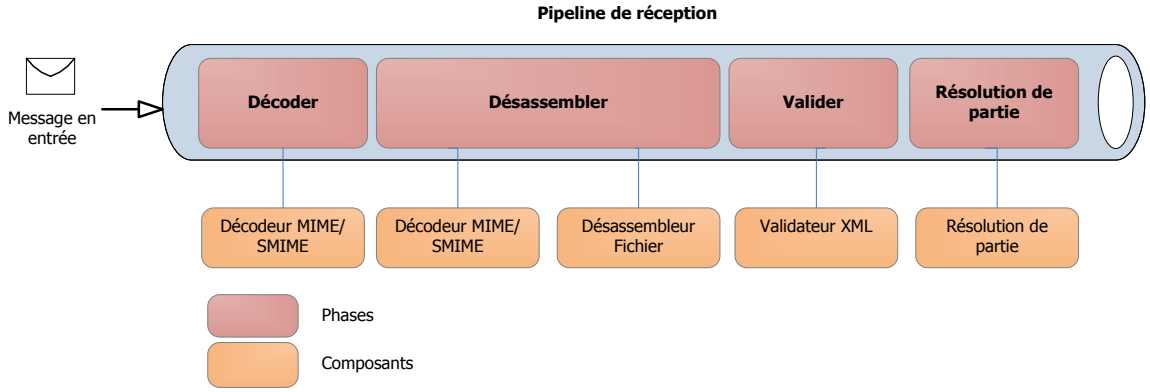

Figure 18 – Pipeline de réception

Les phases d'un pipeline de réception et leurs composants standardisés offerts à chacune d'entre elles sont :

- *Décoder* - BizTalk Server fournit un composant pour cette phase : le *Décodeur MIME/SMIME*. Ce composant peut gérer les messages, ainsi que tous les attachements qu'ils contiennent, qu'ils soient au format MIME ou MIME sécurisé (S/MIME). Le composant convertit les deux types de messages en XML. Il peut également décrypter les messages S/MIME et vérifier leurs signatures digitales.
- *Désassembler* – Deux composants sont ici principalement utilisés. Le *Désassembleur de fichiers* convertit les fichiers en documents XML. Ces fichiers peuvent être de type positionnel, chacun des enregistrements possédant les mêmes longueur et structure, ou de type délimité, ce qui correspond à un format où un caractère désigné est utilisé pour séparer les enregistrements, dans le fichier. Le second composant standard, le *Désassembleur XML*, analyse les messages entrants qui sont déjà au format XML.
- *Valider* - BizTalk Server fournit le composant *Validateur XML* pour cette phase. Comme son nom le suggère, ce composant valide un document XML produit par la phase de désassemblage, par rapport à un schéma ou un groupe de schémas spécifiés. Il retourne une erreur si le document n'est pas conforme à l'un de ces schémas XSD.
- *Résolution de partie* - Le composant fourni pour cette phase, le composant *Résolution de partie*, permet notamment de déterminer l'identité de l'émetteur du message.

La Figure 19 montre les phases et composants d'un pipeline d'émission.

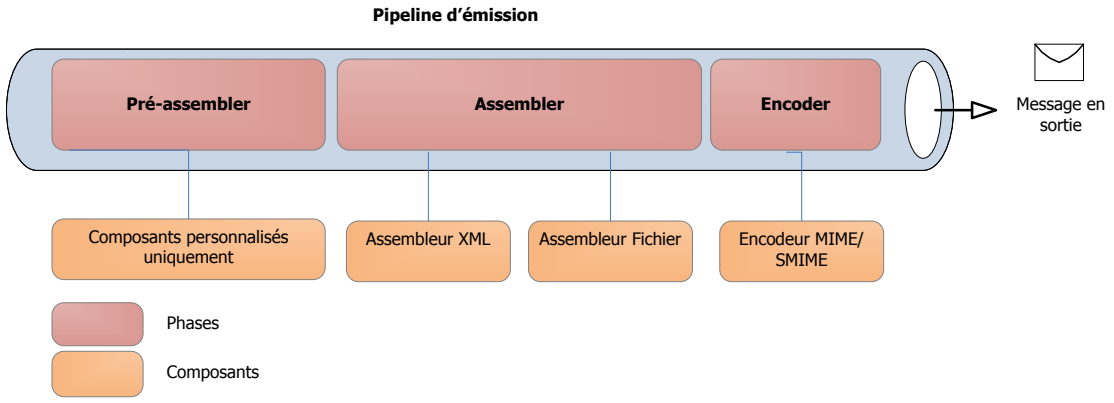

Figure 19 – Pipeline d'émission

Les messages sortants peuvent également passer par de multiples phases, telles que définies par le pipeline d'émission utilisé. Les phases et les composants standardisés d'un pipeline d'émission correspondent aux éléments suivants :

- *Pré-assembler* – Aucun composant par défaut n'est fourni. Au lieu de cela, des composants personnalisés peuvent être insérés selon les besoins.
- *Assembler* – Parallèlement à la phase de désassemblage d'un pipeline de réception, cette phase comporte également principalement deux composants Le composant *Assembleur de fichiers* convertit un message XML en un fichier positionnel ou délimité, tandis que l'*Assembleur XML* permet d'ajouter une enveloppe ou d'apporter d'autres modifications à un message XML sortant.
- *Encoder* – BizTalk Server ne définit qu'un seul composant pour cette phase, l'Encodeur MIME/SMIME. Ce composant conditionne les messages sortants aux formats MIME ou S/MIME. Si le format S/MIME est utilisé, le message peut être également signé numériquement et/ou chiffré.

BizTalk Server définit quelques pipelines par défaut, ceci incluant une simple paire de pipelines émission/réception qui peut être utilisée pour gérer les messages qui sont déjà au format XML. Des pipelines personnalisés peuvent être créés au moyen du *Concepteur de Pipeline*. Cet outil, qui s'exécute au sein de l'environnement Visual Studio 2005 présente une interface graphique qui permet de glisser/déposer des composants, afin de créer des pipelines d'émission et de réception aux comportements requis.

4.2.2.3 Choix des messages avec les souscriptions

Une fois qu'un message a fait son chemin, via un adaptateur et un pipeline de réception, la question suivante qui se pose consiste à savoir où il doit être acheminé. **Une orchestration constitue le plus souvent la cible d'un message, mais il est également possible qu'un message aille directement vers un pipeline d'émission ; le moteur de BizTalk Server 2006 R2 est à la base à un « système de messagerie ».** Dans l'un ou l'autre des cas, la correspondance entre ces messages et leur destination est effectuée via les souscriptions.

Lorsqu'un message est traité par un pipeline de réception, un contexte, contenant les diverses propriétés du message, est généré. Une orchestration ou un pipeline d'émission peuvent souscrire à des messages en fonction des valeurs de ces propriétés. Une souscription ne retourne au souscripteur que les messages qui correspondent aux critères définis par elle. Un message d'entrée peut initialiser un processus métier en réalisant l'instanciation d'une orchestration quelconque ou activer une autre étape au sein d'un processus métier qui est déjà en cours d'exécution. De même, lorsqu'une orchestration envoie un message, ce message est attribué à un pipeline d'émission, en fonction d'une souscription établie par ce pipeline.

4.2.3 Définition d'un processus métier

Envoyer des messages entre des applications hétérogènes est une condition sine qua non à la résolution des problèmes que BizTalk Server adresse. Mais dans la plupart des cas, c'est seulement un moyen d'arriver à une fin : le but réel est de définir et d'exécuter des processus métier en fonction de ces applications : orchestration et coordination de services. Le moteur de BizTalk Server offre deux approches pour réaliser cette opération : les orchestrations et le moteur des règles de gestion.

4.2.3.1 Orchestrations

Comme évoqué en préambule, un processus métier peut être mis en œuvre directement via un langage tel que C#. Mais la création, la maintenance et la gestion de processus métier complexes, au moyen de langages de programmation conventionnels, peuvent représenter un défi. De la même manière, un processus créé de cette façon risque d'apparaître comme un ensemble d'applications disparates, plutôt que comme un processus métier cohérent et coordonné. Ainsi, BizTalk Server 2006 R2 ne retient pas cette approche. Au lieu de cela, il permet la création graphique d'un processus métier. Cette façon de faire peut être plus rapide que de construire le processus directement via un langage de programmation et en facilite la compréhension, l'explication et la modification.

Les processus métier à définir sont, en effet, généralement complexes et incorporent de nombreux contrôles d'intégrité : support de transactions ACID, persistance de l'état

d'interactions à longues durées d'exécution (« *transactions longues* »), mécanismes de compensation et d'exception, opérations imbriquées et parallèles, acquittements ou encore capacités à corréler. Au fur et à mesure de l'accroissement de la complexité du processus, les bénéfices issus d'une conception, d'une implémentation, et d'une documentation au travers d'une méthodologie d'assemblage visuel deviennent évidents. C'est particulièrement le cas lorsque les processus exigent la modification ou la conception de processus qui servent de base à d'autres.

La possibilité de définir des processus métier basés sur des services Web, via l'utilisation de *Web Services Business Process Execution Language* (WS-BPEL ou simplement BPEL, précédemment nommée BPEL4WS pour *Business Process Execution Language for Web Services*) **constitue également un élément clé.**

Ce standard de l'OASIS (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel) développé à l'origine en collaboration par Microsoft, IBM et BEA Systems a été conçu pour orchestrer et coordonner des services Web de façon à ce qu'ils puissant être engagés dans un traitement collaboratif et transactionnel. BizTalk Server en offre aujourd'hui un support limité au travers d'une implémentation « préversion 1.1 ».

Dans le contexte du protocole PRESTO, tandis que les nouvelles spécifications standards des services Web fournissent la méthodologie pour la communication par messages d'application à application et l'invocation de méthodes au-dessus d'un réseau fédéré, ils ne peuvent pas directement en eux-mêmes répondre aux exigences opérationnelles d'un processus métier. Un tel processus est constitué d'un ensemble d'activités dépendantes et ordonnées; son exécution résulte en des résultats prévisibles que l'on peut répéter de façon opportune. BPEL permet aux services Web de répondre finalement à ces exigences. BPEL définit formellement les activités de base et les activités structurées qui sont employées pour composer des processus métier sophistiqués répondant aux besoins et contraintes qui le sous-tendent. Comme un jeu d'instructions BPEL constitue une représentation XML d'un processus avec une structure lexicale et grammaticale précise, il fournit, par là même, un jeu d'instructions lisible et compréhensible pour le documenter exhaustivement.

BPEL est « écrit » en XML Schema et définit formellement les activités de base et les activités structurées qui sont employées pour composer des interactions entre processus métier sophistiqués. Les formes employées dans le *concepteur d'orchestration* de BizTalk Server pour développer ces processus sont des abstractions visuelles de niveau élevé de ces activités. Un processus développé dans le concepteur d'orchestration peut être exporté sous forme d'un document BPEL. Un jeu d'instructions BPEL peut être importé dans n'importe quelle autre application conforme BPEL. Réciproquement, un document BPEL peut être importé dans le concepteur d'orchestration de BizTalk Server afin de produire un diagramme d'orchestration. L'échange normalisé des instructions de processus peut potentiellement faciliter le développement de collaboration de processus métier entre administrations.

S'il est aisé de consommer un service Web ou un service PRESTO existants dans un processus métier, un processus métier peut être publié sous la forme d'un service Web ou d'un service PRESTO et le rendre utilisable par d'autres.

L'exécution des orchestrations est assurée par BizTalk Server. Cette exécution peut être répartie sur plusieurs processus suivant les contraintes de sécurité et eux-mêmes peuvent être répartis sur plusieurs serveurs physiques différents afin de répondre à des besoins d'évolutivité et/ou de montée en charge importante.

La prise en charge par BizTalk Server de l'exécution des orchestrations permet de distribuer leur exécution sur plusieurs instances de processus et sur plusieurs serveurs. Dans le même temps, ces processus métier peuvent être contrôlés facilement, avec un état de fait exploité par la technologie *Business Activity Monitoring* (BAM) qui constitue un environnement général de suivi des activités d'un processus métier particulier.

R2 introduit à ce titre la notion d'intercepteurs BAM pour la composante Windows Workflow Foundation[18] (WF) de la version 3.0 du Framework .NET. Ces intercepteurs permettent à des évènements et à des données au sein d'un workflow WF d'être capturés et délivrés à BAM autorisant ainsi des scénarios d'instrumentation cross BizTalk et WF.

4.2.3.2 Moteur des règles de gestion

BizTalk Server fournit le moteur des règles de gestion permettant de créer et de modifier des jeux de règles de gestion.

Afin de mieux percevoir l'utilité d'une telle approche, il convient de penser à ce que requiert la modification d'une règle de gestion, implémentée au sein d'une orchestration. Il faut ouvrir l'orchestration dans Visual Studio 2005, modifier les formes appropriées (voir les composants.NET qu'elle invoque), puis, construire et déployer l'assemblage modifié. Ceci requiert également l'arrêt et le redémarrage de l'application BizTalk Server qui utilise cette orchestration. Si, au lieu de cela, cette règle de gestion est mise en œuvre au moyen du moteur des règles de gestion, elle peut être modifiée sans recompiler ni relancer quoi que ce soit. Tout ce qu'il est nécessaire de faire est d'utiliser l'*éditeur des règles de gestion* pour modifier la règle désirée, puis de déployer le nouveau jeu de règles. La modification prendra effet immédiatement.

La définition d'un jeu de règles de gestion suppose la définition préalable d'un vocabulaire nécessaire pour spécifier ces règles. Chaque terme du vocabulaire correspond au nom explicite d'une information quelconque et peut être initialisé en tant que constante, via l'attribut d'un schéma XML quelconque (et ainsi d'un message entrant), le résultat d'une requête SQL exécutée sur une base de données quelconque, ou même une valeur contenue dans un composant .NET.

Une fois un tel vocabulaire défini, des politiques publiques ou de services publics qui emploient ce vocabulaire peuvent être créées. Chaque politique peut contenir une ou plusieurs règles de gestion. Une règle utilise les termes définis dans un vocabulaire quelconque, en association avec des opérateurs logiques, pour déterminer le comportement d'un processus métier. Une règle de gestion peut définir la façon dont les valeurs contenues dans un document XML de réception peuvent affecter les valeurs créées dans un document XML d'envoi, ou définir quelle valeur contenue dans une base de données, seront affectées par ces valeurs reçues, etc.

Afin d'exécuter une politique, une orchestration peut faire appel à une forme *CallRules*, qui crée une instance du moteur des règles de gestion, identifie quelle politique exécuter, puis lui passe les informations dont elle a besoin, tel qu'un document XML reçu. Le moteur des règles de gestion peut également être invoqué dans un programme, via un modèle objet basé sur .NET, ce qui lui permet d'être appelé potentiellement par d'autres modules d'une plateforme de services.

Les vocabulaires et règles de gestion peuvent être bien plus complexes – et bien plus puissants – que les exemples simples décrits ici. Mais l'idée maîtresse consistant à définir un vocabulaire, puis à définir des jeux de règles qui utilisent ce vocabulaire, est au cœur du moteur des règles de gestion. L'objectif est d'offrir un moyen simple de créer et de travailler avec les règles qui définissent les processus métier.

Une orchestration synthétise le flux d'un processus métier en l'exprimant graphiquement plutôt qu'avec du code. De même, le moteur des règles de gestion permet d'exprimer les règles d'un processus métier selon un niveau d'abstraction plus élevé. Ensemble, ces deux technologies offrent une approche efficace pour créer la logique fonctionnelle d'exécution des Hubs de messages et d'intégration.

Venons à présent au support du protocole PRESTO en tant que tel.

[18] WF comprend une bibliothèque *BPEL Activity Library* qui implémente les concepts définis par la version 1.1 de la spécification BPEL. Cette dernière permet la création de *workflows* qui peuvent être exportés vers BPEL De même, une définition BPEL peut être importée au sein de WF pour la création d'un *workflow*.

4.2.4 Kit de démarrage Adaptateurs pour BizTalk Server 2006 R2

En collaboration avec la société CODit (http://www.codit.eu), Microsoft France a récemment mis à disposition le Kit de démarrage Adaptateurs pour BizTalk Server 2006 R2 en complément du Kit de démarrage PRESTO (Cf. section § 3.10 « DOCUMENTATION ET IMPLEMENTATIONS DISPONIBLES »).

Ce kit de démarrage également publié sous le contrat de licence de logiciel libre CeCILL-B[19] comprend un ensemble d'adaptateurs PRESTO reposant sur la pile de communication Windows Communication Foundation (WCF) du .NET Framework 3.0.

La version 1.1 du protocole PRESTO [PRESTO-Ref] est supportée par la version courante du Kit de démarrage.

Ces adaptateurs sont les suivants:

- *Adaptateur PRESTO interne* – Cet adaptateur destiné à l'émission et à la réception de messages PRESTO s'exécute directement dans l'instance BizTalk considérée ; celle-ci est créée depuis la console d'administration de Biztalk Server ; plusieurs instances peuvent être définies et le choix de l'instance se fait lors du paramétrage d'un port BizTalk.

  Cet adaptateur permet à l'utilisateur de sélectionner les différentes options offertes du protocole PRESTO et de configurer, si nécessaire, les attributs de la liaison (*binding*) WCF personnalisée associée qui se conforme aux caractéristiques du protocole PRESTO ainsi que les comportements (*behavior*) et les extensions de comportement pour un emplacement de réception (*receive location*) et un port d'émission (*send port*).

- *Adaptateur PRESTO isolé* – Cet adaptateur destiné uniquement à la réception de messages PRESTO s'exécute le code dans un hôte isolé, comme Microsoft Internet Information Server (IIS) par exemple.

  Cet adaptateur permet à l'utilisateur de sélectionner les différentes options offertes du protocole PRESTO et de configurer, si nécessaire, la liaison (*binding*) WCF personnalisée associée qui se conforme aux caractéristiques du protocole PRESTO ainsi que les comportements (*behavior*) et les extensions de comportement pour un emplacement de réception (*receive location*) s'exécutant au niveau d'un hôte isolé.

Ces deux adaptateurs permettent à Microsoft BizTalk Server 2006 R2 de recevoir et d'envoyer des messages via le protocole PRESTO en ajoutant un transport PRESTO spécifique à la plateforme BizTalk Server.

[19] Cf. http://www.cecill.info/licences/Licence_CeCILL-B_V1-fr.txt.

La Figure 20 illustre l' « anatomie » de ces adaptateurs.

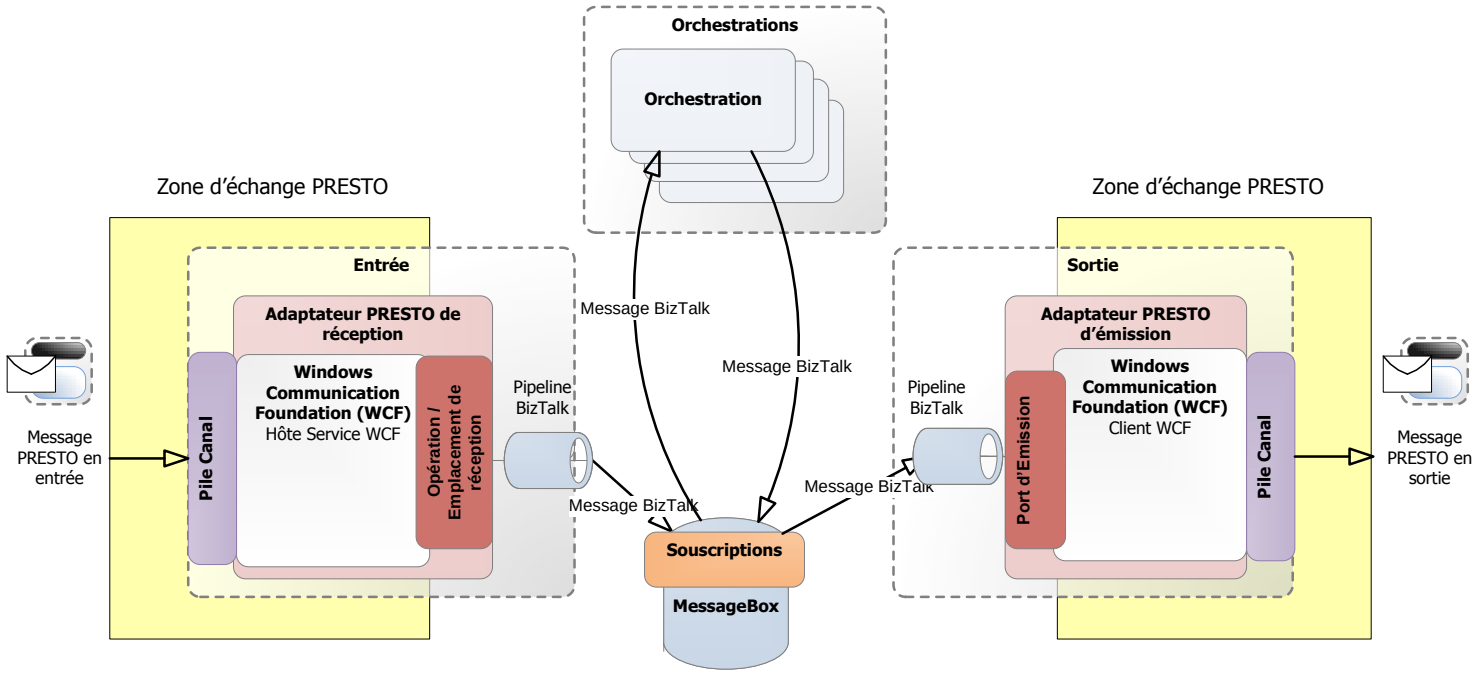

Figure 20 – Adaptateurs PRESTO pour BizTalk Server 2006 R2

Ces deux adaptateurs s'accompagnent des assistants de publication (*BizTalk PRESTO Service Publishing Wizard*) et de consommation (*BizTalk PRESTO Service Consumming Wizard*). L'assistant de publication peut être utilisé pour créer et publier des orchestrations BizTalk en tant que mandataire récepteur PRESTO, et de publier les schémas PRESTO associés, au même titre que de publier un point d'échange de métadonnées dans IIS.

La notion d'échange de métadonnées n'est pas couverte dans la version 1.1 du protocole PRESTO. Ceci doit faire l'objet d'une prochaine révision.

Le Kit de démarrage comprend également le code source C# des adaptateurs PRESTO (et assistants associés) de façon à ce que ces derniers soient, le cas échéant, modifiés et recompilés en vue d'un usage plus spécifique du protocole PRESTO.

Ce kit de démarrage est disponible à l'adresse Internet http://www.microsoft.com/downloads/details.aspx?FamilyID=826D4D2D-8E8C-439C-8104-B6DB89EEE626.

Ce kit est également disponible sur AdmiSource, la plateforme collaborative proposée à l'ensemble des administrations Françaises pour leurs développements de logiciels libres, à l'adresse internet : http://admisource.gouv.fr/projects/presto.

Annexe A. Références

Cette annexe regroupe des liens vers des compléments d'informations sur les standards sur lesquels repose le « PRotocole d'Échanges Standard et Ouvert » (alias PRESTO).

[PRESTO-Guide]
"Guide pour le support de PRESTO", juin 2006.

[PRESTO-Ref]
"Protocole d'échanges entre acteurs de l'administration électronique", juin 2006.

[MTOM]
"SOAP Message Transfer Optimization Mechanism", janvier 2005.

[SOAP 1.2]
"SOAP Version 1.2 Part 1: Messaging Framework," juin 2003.

[WSDL 1.1]
"Web Service Description Language (WSDL) 1.1", mars 2001.

[WS-AddressingAugust2004]
"Web Services Addressing (WS-Addressing)", août 2004.

[WS-Addressing10]
Web Services Addressing (WS-Addressing)", mai 2006.

[WS-MetadataExchange]
"Web Services Metadata Exchange (WS-MetadataExchange)", septembre 2004.

[WS-Policy]
"Web Services Policy Framework (WS-Policy)", septembre 2004

[WS-RM1.0]
"Web Services Reliable Messaging (WS-ReliableMessaging) 1.0", février 2005.

[WS-RM1.1]
"Web Services Reliable Messaging (WS-ReliableMessaging) 1.1", juin 2007.

[WS-RMPolicy]
"Web Services Reliable Messaging Policy Assertion (WS-RM Policy) 1.1", juin 2007.

[WS-Security]
"Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", mars 2004.

[WS-SecX509]
"Web Services Security: X.509 Token Profile V1.0", mars 2004.

[WS-SecurityPolicy]
"Web Services Security Policy Language (WS-SecurityPolicy)", avril 2006.

[XMLDSIG]
“XML-Signature Syntax and Processing”, mars 2002.

[XMLENC]
“XML Encryption Syntax and Processing”, août 2002.

[XML Schema, Part 1]
"XML Schema Part 1: Structures", mai 2001.

[XML Schema, Part 2]
"XML Schema Part 2: Datatypes", mai 2001.

[XOP]
"XML-binary Optimized Packaging (XOP) 1.0", janvier 2005.

Annexe B. Pour des informations complémentaires

http://www.microsoft.com/biztalk

Pour les dernières informations sur Windows Server System, vous pouvez consulter le site dédié Windows Server System à l'adresse Internet http://www.microsoft.com/windowsserversystem.