Published on *Marco de Desarrollo de la Junta de Andalucía* (http://127.0.0.1/servicios/madeja)

Integración

Código: ARQ_INT

El área de Integración contiene pautas para el desarrollo de aplicaciones con arquitectura orientada a servicio, con la finalidad de aumentar el grado de interoperabilidad de los sistemas de información y con la capacidad de atender de forma más eficiente los procesos de negocio.

Se recogerán las recomendaciones de la Plataforma de Interoperabilidad de la Junta de Andalucía (PLATINA) y una propuesta tecnológica de referencia, agilizando el desarrollo de nuevos servicios y el uso de los existentes.

En el futuro se añadirá el catálogo de servicios disponibles en la Junta de Andalucía, para dar una descripción funcional y procedimental de su uso, registro de usuarios y recomendaciones.

Además de las indicaciones dadas en este subsistema, se ha creado un área específica en Desarrollo, Seguridad, Servicios Web donde se resumen las recomendaciones sobre el uso y diseño de servicios web seguros.

Código	Título	Tipo	Carácter
LIBP-0006	Creación de Servicios Web	Directriz	Obligatoria
PAUT-0031	Identificación de Mensajes y EndPoints (WS-Addressing)	Directriz	Obligatoria
PAUT-0033	Manejo de Errores	Directriz	Obligatoria
PAUT-0030	Política de Versionado	Directriz	Obligatoria
LIBP-0007	Reglas de Codificación	Directriz	Obligatoria
LIBP-0321	Uso de Apis y Frameworks de Servicios Webs	Directriz	Obligatoria
LIBP-0320	Uso de especificaciones y estándares de servicios web	Directriz	Recomendada

Código	Título	Tipo	Carácter
RECU-0100	Especificaciones y estándares de servicios web	Especificación	Recomendado
RECU-0019	Plataforma de Interoperabilidad de la Junta de Andalucía	Ficha	Recomendado

Creación de Servicios Web

- **Área:** Integración
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: LIBP-0006

La recomendación de MADEJA puede resumirse en tres puntos:

- Modelo de Mensajes o Modelo SOA
- Aproximación Contract-First
- Estilo de codificación document/literal

En la actualidad. los Servicios Webs pueden ser desarrollados siguiendo una gran variedad de aproximaciones que incluyen el uso de diversas técnicas y herramientas, cada una de ellas con sus ventajas e inconvenientes.

El objetivo de MADEJA en este apartado es plasmar las distintas posibilidades existentes a la hora de crear un servicio web, recomendando aquellas que más ventajas aportan a la Junta de Andalucía.

Pautas

Título	Carácter
Modelo RPC, SOA y REST	Obligatoria
Code-first y Contract-First	Obligatoria
Estilos de codificación RPC/encoded, RPC/literal y document/literal	Obligatoria

Modelo RPC, SOA y REST

MADEJA recomienda la creación de Servicios Web según el modelo de Mensajes o Modelo SOA, en lugar de los modelos RPC o REST

Tipo de Modelo	Carácter	Observaciones
Modelo de Mensajes (SOA)	Recomendado	Obligatorio para servicios que hacen uso de PLATINA
Modelo RPC	Permitido	Para servicios internos que NO hacen uso de PLATINA
Modelo REST	Permitido	Para servicios internos que NO hacen uso de PLATINA

Dentro del mundo de los Web Services, tres modelos de programación conviven en la actualidad; el Modelo RPC, el Modelo de Mensajes (Modelo SOA) y el modelo REST.

Dentro del marco de los proyectos de la Junta de Andalucía, ha sido muy común hasta ahora el desarrollo de Servicios Web según el modelo RPC. La introducción de PLATINA, como plataforma de interoperabilidad dentro de la Junta de Andalucía, trae consigo nuevas necesidades a la hora de crear Servicios Web. Los servicios web publicados dentro de PLATINA necesitan ser muy flexibles, fácilmente versionables y poseer un alto grado de interoperabilidad. Todo esto nos ha llevado a recomendar dentro de MADEJA la creación de Servicios Web según el modelo de Mensajes o Modelo SOA, en lugar de los modelos RPC o REST.

Esta recomendación, tiene un carácter obligatorio para todo servicio Web que se integre sobre PLATINA. En el caso de Servicios Web que NO se integren dentro de PLATINA, estos podrán ser creados con cualquiera de los tres modelos de programación, aunque nuestra recomendación es que estos se hagan según el Modelo SOA en detrimento del modelo RPC y REST.

Volver al índice

Code-first y Contract-First

La recomendación de MADEJA es que los Servicios Webs se desarrollen siguiendo la aproximación Contract-first

Tipo de Aproximación	Carácter	Observaciones
Aproximación Contract-first	Recomendado	-
Aproximación Code-first	Permitido	Para servicios internos que NO hacen uso de PLATINA

A la hora de desarrollar un Servicio Web el programador actual puede optar por desarrollarlos siguiendo la aproximación "Code-First" o la aproximación "Contract-First".

En la aproximación "Code-First", primero se codifica el código del servicio dentro del lenguaje de programación y después se genera el WSDL describiendo el servicio de forma automática, a través del framework de Servicios Webs elegido.

Esta opción, hasta la actualidad, ha sido la más seguida dentro de la Junta de Andalucía, dada su simplicidad y rapidez a la ahora

de crear Servicios Webs. Sin embargo, esta opción presenta una serie de inconvenientes que desaconsejan su uso. El contrato del servicio está muy ligado a la implementación del mismo, de forma que un cambio en dicha implementación puede conllevar cambios en el WSDL. Además, el contrato está también muy ligado al framework de programación empleado, de forma que un cambio de framework o una actualización de versión del mismo, pueden conllevar cambios en el WSDL, con las consiguientes implicaciones en los consumidores de dicho servicio.

En la opción "Contract-First", la idea es la contraria, primero se genera el WSDL y después se codifica el servicio. Esta opción carece de los inconvenientes de la opción anterior y aporta numerosas ventajas a la hora de desarrollar y mantener los Servicios Webs.

Por todo ello, la recomendación de MADEJA es que los Servicios Webs se desarrollen siguiendo la aproximación Contract-first. Esta recomendación toma carácter obligatorio para todos aquellos servicios web que se expongan dentro de PLATINA, siendo recomendable, pero no obligatorio, su adopción para los restantes Servicios Webs desarrollados dentro de la Junta de Andalucía.

Volver al índice

Estilos de codificación RPC/encoded, RPC/literal y document/literal

La codificación recomendada dentro de MADEJA es la codificación document/literal

Tipos de Codificación	Carácter	Observaciones
Estilo document/literal	Recomendado	Obligatorio para servicios que hacen uso de PLATINA
Estilo RPC/Encoded	No recomendado	No soportado para WS-I
Estilo RPC/literal	Permitido	Para servicios internos que NO hacen uso de PLATINA

En la actualidad, existen tres mecanismos para la codificación de los mensajes SOAP: RPC/encoded, RPC/literal y Document/literal.

La codificación RPC/encoded no está recomendada por la WS-I y queda prohibido su uso.

La codificación RPC/literal no está aconsejada por MADEJA, aunque se permitirá su uso en aquellos Servicios Webs que se desarrollen siguiendo un modelo RPC y que no se integren dentro de PLATINA.

La codificación document/literal es la recomendada dentro de MADEJA, siendo obligatoria para Servicios Web que se integren dentro de PLATINA.

Volver al índice

Identificación de Mensajes y EndPoints (WS-Addressing)

- **Área:** Integración
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: PAUT-0031

La especificación WS-Addressing debe ser recogida, de forma obligatoria, por todos los servicios Webs (RPC y SOA) desarrollados para la Junta de Andalucía. El elemento **MessageID** definido por esta especificación como opcional pasará a ser obligatorio dentro de MADEJA para todas las aplicaciones.

La identificación de los mensajes SOAP y EndPoints involucrados en los procesos de comunicación por Servicios Web es un elemento esencial a la hora de realizar procesos de auditoría, monitorización, detección de duplicados, etc. Además, estos mismo elementos posibilitan la creación de un gran número de procesos de enrutamiento y mediación basados en los mismos.

La especificación WS-Addressing recoge mecanismos para la identificación de estos elementos de forma estandarizada e independiente al protocolo de transporte empleado, mediante la definición de un conjunto de cabeceras SOAP. En la actualidad, esta especificación cuenta con implementaciones en la mayoría frameworks de Servicios Webs, lo que anima, aún más, a su recomendación dentro de MADEJA.

Manejo de Errores

- **Área:** Integración
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: PAUT-0033

Resumen de la pauta:
* La notificación de errores dentro de los Servicios Webs deberá realizarse mediante el empleo de SOAP Faults.
* Siempre que sea posible, los mensajes SOAP Faults seguirán la estructura definida en la especificación SOAP 1.2 en lugar de la recogida en SOAP 1.1
* Se deberá evitar incluir dentro de los mensajes de error información sensible del servicio, detalles de la implementación del mismo e información no útil, de cara a los consumidores, dentro de los mensajes de error.

Tipo de Errores

Dentro de toda llamada por Servicios Webs se pueden dar tres tipos básicos de errores:

- Errores en la invocación del servicios.
- Errores específicos del servicio.
- Warnings.

Los errores en la invocación del servicio, corresponden con errores independientes a los consumidores y productor del servicio que afectan a cuestiones de protocolos, comunicaciones, frameworks, etc. Estos errores son capturados y notificados automáticamente por los framework de Servicios Webs empleado, generando normalmente, mensajes del tipo SOAP Fault para notificar los mismos. Este tipo de errores, dado su carácter genérico no son incluidos dentro del WSDL del servicio.

Los errores específicos del servicio, pueden ser notificados mediante mecanismos genéricos y estandarizados; como los SOAP Faults, o mediante mecanismos propietarios de las aplicaciones como; mensajes o campos específicos de error. Dentro de la junta, ha sido muy habitual la notificación de los errores dentro de campos creados para tal efecto dentro de las respuestas de WS.

La recomendación de MADEJA es que los errores de aplicación se notifiquen mediante el empleo de SOAP Faults, en lugar de mecanismos particulares de cada aplicación. Los SOAPs Fault generados por un servicio deberán ser recogidos dentro del WSDL del servicio.

Dentro de la ejecución de un servicio, se pueden dar casos en los que se detecten errores que no impidan la ejecución del mismo, estos errores son conocidos como warnings. En ciertos servicios puede ver conveniente notificar estos warnings a los consumidores de los servicios empleando soluciones particulares en cada caso. Madeja no definirá, por el momento la forma de notificación de warnings en los servicios, dejando que cada servicio elija la implementación o no de dichos mecanismos.

Soap Fault

El mecanismo estándar para la definición de los errores dentro de un Servicio Web es el SOAP Fault. Las etiquetas Soap fault varían entre las distintas especificaciones de SOAP 1.1 y SOAP 1.2.

La recomendación de madeja es que se siga, siempre que sea posible, la definición de SOAP Fault recogida en la especificación SOAP 1.2 en lugar de la recogida en SOAP 1.1.

En los casos en los que no sea posible y se siga la versión recogida en SOAP 1.1, se deberá seguir las restricciones puestas en la WS-I, evitándose el empleo de dot Code.

Recomendaciones básicas para Errores del Servicio

- Los errores han de ser **comunicados de forma independiente** al S.O., lenguaje de programación y aplicación.
- Los errores han de ser **presentados de tal forma que se facilite su interpretación y procesado**. Para cumplir este concepto se recomienda la utilización de códigos y subcódigos de error.
- **Nunca exponer detalles internos de implementación** del servicio a los consumidores, ya que esto supone un riesgo de seguridad para el propio servicio y aporta información NO Útil de cara al consumidor del mismo.
- **Nunca devolver la traza de las excepciones**. La traza de las excepciones aporta mucha información no útil para el consumidor y expone detalles de implementación del servicio.
- **Sólo los errores tratables por el consumidor**, por ejemplo los referentes a errores en los parámetros de entrada del servicio, **deben ser notificados de forma detallada**. Errores no procesables por el consumidor, como pej. la caída de una B.D., deberán ser enmascarados y transmitidos mediante un error genérico ("pej. Error interno").

Política de Versionado

- **Área:** Integración
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: PAUT-0030

La gestión de versiones en Servicios Web es fundamental para garantizar un funcionamiento correcto al hacer uso de ellos.

Tipo	Trazabilidad	Pauta
RPC	-	El elemento empleado para dirigir la política de versionado es el namespace del servicio, al cual se le dotará de un número de versión
SOA	SIN	Esta aproximación sería similar a la del modelo RPC. El namespace del servicio incluirá un número de versión, el cual no variará ante los cambios con compatibilidad hacia atrás, y que se incrementaría en caso contrario.
SOA	CON	En esta aproximación se asignará un namespace con versión al Servicio Web. A diferencia de los casos anteriores, este namespace sólo será empleado para definir al servicio, no será reutilizado por otros elementos como los types, bindigs, etc. Al igual que en los otros casos, cuando se produzca un cambio que implique perdida en la compatibilidad hacia atrás, el número de versión del servicio se incrementará para obligar así a regenerar las interfaces en los consumidores del servicio.
REST	-	Sin recomendación para el mantenimiento de la versión

Introducción

Al igual que cualquier otro componente software, los Servicios Webs no son elementos inmutables en el tiempo, sino que están sujetos a cambios a lo largo de toda su vida. El componente distribuido de los Servicios Web hace que los cambios en su interfaz puedan repercutir en los consumidores, obligándoles incluso a reimplementar su interfaz con el servicio. Todo esto marca la necesidad de definir una política de versionado dentro de los Servicios Webs, que aborde la creación de reglas que regulen el versionado de los Servicios Web y sus componentes.

El versionado dentro de los Servicios Webs es un tema abierto en la actualidad pero no nuevo, como lo muestran las numerosas especificaciones y recomendaciones que abordan los problemas de versionado dentro de los Servicios Webs, XMLs y Esquemas.

Toda política de versionado de Servicios Webs debe identificar los tipos de cambios que se realizan en un servicio y tener en cuenta el impacto de los mismo. Los cambios normalmente se agrupan en; cambios con compatibilidad hacía atrás y sin compatibilidad hacia atrás.

Los cambios con compatibilidad hacia atrás, son cambios en el Servicio Web que no "rompen" el servicio y que por tanto, no obliga a reimplementar los consumidores de dicho servicio. Ejemplos de estos cambios son:

- Inclusión de nuevas operaciones.
- Inclusión de nuevos tipos dentro del esquema.

Los cambios sin compatiblidad hacia atrás, son cambios que si "rompen" el servicio y que por tanto, obligan a reimplementar los consumidores de dicho servicio. Ejemplos de estos cambios son:

- Eliminación de una operación.
- Renombrado de una operación.
- Cambio en los parámetros de una operación.
- Cambios en un tipo de datos existente dentro del esquema.

Definición de la Política

MADEJA permite la creación de servicios Webs según tres modelos: RPC, SOA y REST. Las políticas de versionado variarán en cada caso para adaptarse a las posibilidades que proporciona cada modelo.

Política de Versionado en RPC

En el modelo RPC, los elementos que definen un Servicio Web, están todos incluidos dento de un WSDL generado por el framework, lo que dificulta la personalización del mismo.

En este caso, el elemento empleado para dirigir la política de versionado es el namespace del servicio, al cual se le dotará de un número de versión (major.minor).

En el caso de que se produzca un cambio en la interfaz del servicio y se mantenga la compatibilidad hacia atrás, el namespace del servicio no se cambiará.

En caso que se produzca un cambio que si afecte a la compatibilidad hacia atrás de servicio, se deberá incrementar el número

de versión del namespace, lo que "romperá" a los consumidores del servicio, obligándoles a reimplementar su interfaz con él.

La ventaja de esta política de versionado es su facilidad de implementación. Por contra, su principal desventaja es que la solución no permite registrar los cambios producidos dentro de la interfaz de Servicio Web, es decir no hay trazabilidad en los cambios.

Política de Versionado en SOA

Al contrario que en el modelo RPC en el modelo de mensajes (SOA), el WSDL se crea a mano y es fácilmente personalizable, lo que posibilita un gran número de posibilidades a la hora de definir la política de versionado.

En este apartado se dan dos políticas a elección de la Dirección de Proyecto, una sin capacidad de trazabilidad de los cambios y de fácil adopción, y otra con capacidad de trazabilidad ante cambios pero de mayor complejidad.

Aproximación SIN Trazabilidad

Esta aproximación sería similar a la del modelo RPC. El namespace del servicio incluirá un número de versión (major.minor), el cual no variará ante los cambios con compatibilidad hacia atrás, y que se incrementaría en caso contrario.

En esta aproximación todos los elementos del wsdl (bindings, types, etc) estarán definidos bajo el mismo namespace del servicio.

Si el servicio se compone de varios ficheros (xsds, wsdls, etc), estos incluirán el numero de versión del servicio.

Aproximación CON Trazabilidad

En esta aproximación se asignará un namespace con versión (major.minor) al Servicio Web. A diferencia de los casos anteriores, este namespace sólo será empleado para definir al servicio, no será reutilizado por otros elementos como los types, bindigs, etc. Al igual que en los otros casos, cuando se produzca un cambio que implique perdida en la compatibilidad hacia atrás, el número de versión del servicio se incrementará para obligar así a regenerar las interfaces en los consumidores del servicio.

A cada tipo de elemento incluido dentro del WSDL (types, bindings, interfaces) se le asignará un namespace propio con un número de versión independiente. A diferencia de los casos anteriores, estos namespaces podrán usar en su número de versión los tres digitos ((major.minor.revision).

La siguiente tabla incluye la recomendación de namespaces para cada elemento.

Elemento	Namespace	Comentarios
Servicio Web	urn:juntadeandalucia:(consejería):(aplicación):(servicio -opc):(version)	-
Bindings	urn:juntadeandalucia:(consejería):(aplicación):(servicio -opc):bind:(version)	-
Tipos	urn:juntadeandalucia:(consejería):(aplicación):(servicio -opc):type:(nombre esquema -opc):(version)	-
Interfaz	urn:juntadeandalucia:(consejería):(aplicación):(servicio -opc):intf:(version)	-

Cada vez que un elemento nuevo se añada al servicio, los elementos antiguos se mantendrán en su correspondiente namespace y los nuevos incluirán en su namespace el número de versión actualizada.

Por ejemplo, si se añade dos nuevas interfaces, el wsdl incluiría a las anteriores en el mismo espacio de nombre

```
urn:juntadeandalucia:cica:APLICACION:SERVICIO:intf:v1.0.0
```

e incluiría a las nuevas en un nuevo espacio de nombres actualizado,

```
urn:juntadeandalucia:cica:APLICACION:SERVICIO:intf:v1.0.1
```

de esta forma es posible tracear los cambios realizados dentro del WSDL.

Si el cambio dentro del WSDL implica un cambio dentro de un elemento existente, se incrementará la versión del namespace al que pertenece el elemento modificado. El cambio de versión del namespace también afectará a todos los elementos que comparten el mismo namespace que el elemento modificado. Ya que el cambio de un elemento es un cambio que rompe con la compatibilidad del servicio, el namespace del servicio deberá incrementarse también para notificar este hecho.

Si la definición del servicio se ha dividido en varios archivos, cada archivo deberá contener el número de versión del tipo de componente incluido.

Por ejemplo, el archivo interfaz.wsdl, incluye interfaces de la versión 1.0, el nombre del fichero se debería renombrar a interfaz_v1.0.wsdl

---

Reglas de Codificación

- **Área:** Integración
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: LIBP-0007

Los ficheros WSDLs, esquemas XSD y mensajes SOAPs intercambiados dentro de un Servicio Web deben seguir las normas de codificación y formatos especificados en MADEJA.

Pautas

Título	Carácter
UTF-8	Obligatoria
Codificación de los Namespaces	Obligatoria
Codificación de los WSDLs	Recomendada

UTF-8

Los ficheros WSDLs, esquemas xsd y mensajes SOAPs intercambiados dentro de un Servicio Web, deberán estar codificados en UTF-8

Volver al índice

Codificación de los Namespaces

Los namespaces dentro de los WSDLs y esquemas empleados dentro de los Servicios Web deben seguir el formato normalizado por MADEJA, el cual incluye una referencia a la Junta de Andalucía, consejería, aplicación, servicio, componente y número de versión.

La definición de los namespaces dentro de los Servicios Webs es un tema muy importante a tener en cuenta. Una correcta definición de los mismos favorece la reutilización y catalogación de los servicios, da soporte a los procesos de versionado y posibilita la creación de cierto tipo de servicios horizontales de enrutación y mediación.

Los namespaces dentro de los WSDLs y esquemas empleados dentro de los Servicios Web, se normalizan dentro de MADEJA.

Para la definición de los namespaces se puede optar por emplear URIs basadas en URLs o urn. El formato recomendado por MADEJA es el de urn, al ser más claro y requerir menos procesos de normalización que el de URLs.

El formato recomendado es el presentado a continuación.

Formato
urn:juntadeandalucia:(consejería):(aplicación):(servicio -opc):(componente -opc):versión

Descripción de los campos

Elemento	Obligatorio	Descripción	Comentarios
juntadeandalucia	SI	Referencia a la Junta de Andalucía	-
Consejería	SI	Referencia a la consejería impulsora del servicio	-
Aplicación	SI	Nombre de la aplicación a la que pertenece el servicio	-
Servicio	NO	Nombre del servicio	-
Componente	NO	Nombre del elemento o componente	-
Versión	SI	Número de versión	-

Ejemplos

```
urn:juntadeandalucia:cice:eco:ecobus:v1.1

urn:juntadeandalucia:cice:g3:pdcsrv:bind:v1.0

urn:juntadeandalucia:cice:g3:pdcsrv:intf:v1.1

urn:juntadeandalucia:cice:g3:pdcsrv:types:v1.1
```

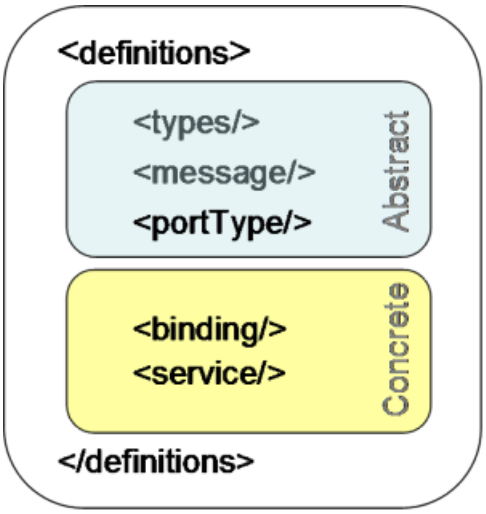
Volver al índice

Codificación de los WSDLs

Para el caso de Servicios Webs codificados según la aproximación Contract-first en MADEJA se recomienda separar las dos partes lógicas de un WSDL (parte abstracta y parte concreta) en dos o más ficheros WSDLs y XSDs.

Un WSDL puede ser dividido en dos partes; la parte abstracta y la parte concreta.

La parte abstracta está formada por los elementos <types>, <message> y portType (o <interface> en 2.0), mientras que a la parte concreta, le corresponden los elementos <binding> y <service>.



Para el caso de Servicios Webs codificados según la aproximación Contract-first en MADEJA se recomienda separar ambas partes lógicas en dos o más ficheros WSDLs y XSDs. El objetivo principal de esta división es favorecer la reutilización de los distintos elementos que componen el servicio web y facilitar las operaciones de versionado sobre los mismos.

Por ejemplo, la parte abstracta se podría dividir en otras dos partes, una compuesta por ficheros xsd donde se recogen los elementos type, y otra compuesta por un fichero WSDL, que importa los xsd anteriores y donde se definirían los elementos messages y portType. La parte concreta estaría formada por otro fichero wsdl que importaría al WSDL anterior y donde se definiría los elementos bindings y services del Servicio Web.

Se puede encontar un ejemplo real en la especificación WSRP

En el caso de que el Servicio Web incluyan elementos como seguridad, Addressing, etc, el WSDL incluirá las políticas empleadas según se define en la especificación WS-Policy.

Volver al índice

Uso de Apis y Frameworks de Servicios Webs

- **Área:** Integración
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Obligatoria

Código: LIBP-0321

Se indica qué APIs y Frameworks para servicios web son recomendados y cuales no

Pautas

Título	Carácter
Metro (WSIT)	Recomendada
Axis1	No Recomendada
Axis2	Recomendada
XFire	No Recomendada
Apache CXF	Recomendada
Spring Web Services	Recomendada

Metro (WSIT)

Se recomienda el uso de Metro (WSIT)

Es un paquete de diferentes tecnologías, entre las que se encuentran **JAX-WS**, **JAXB**, y **WSIT** (implementación de webservices, implementación de XML-Bindings y Webservices intercomunication tecnology, que permite comunicar sin problemas webservices java y .net 3)

Es la actualización del antiguo JWSDP (Java Web Services Developer Pack). Está incluido en Glassfish (Servidor de aplicaciones de Sun) a partir de su versión 2.

Volver al índice

Axis1

No se recomienda el uso de Axis1

Apache Axis es una implementación OpenSource de SOAP que proporciona un entorno de ejecución para Servicios Web implementados en Java

Entre otras cosas Axis proporciona:

- Un entorno de ejecución para Servicios Web Java (*.jws)
- Herramientas para crear WSDL desde clases java.
- Herramientas para crear clientes Java desde un WSDL.
- Herramientas para desplegar, probar y monitorizar Servicios Web.
- Integración con servidores de aplicaciones y contenedores de Servlets.

Volver al índice

Axis2

Se recomienda el uso de Axis2

La nueva arquitectura, en la que se basa Axis2, es más flexible, eficiente y configurable en comparación con la de Axis1.x. Ciertos conceptos probados y bien establecidos de Axis 1.x, como *handlers* etc., se han conservado en la nueva arquitectura.

Apache Axis2 no sólo soporta SOAP 1.1 y SOAP 1.2, sino que también integra soporte para el estilo REST para servicios web y Spring Framework.

Volver al índice

XFire

No se recomienda el uso de XFire

XFire es un framework para desarrollar Servicios Web SOAP en Java a través de la sencillez de uso de su API soportada por los

principales estándares. Este entorno, al igual que el Apache Axis, utiliza componentes del proyecto GlassFish. Sigue la filosofía POJO (Plain Old Java Object), en la que cualquier clase puede ser un Servicio Web, no se necesita extender ninguna clase ni implementar ningún interface para crear un Servicio Web.

Actualmente el proyecto XFire como tal no sigue activo, sino que continua como "Apache CXF"

Volver al índice

Apache CXF

Se recomienda el uso de Apache CXF

Es un framework de servicios de software libre. CXF nos ayuda a construir y desarrollar servicios utilizando JAX-WS como API de programación. Estos servicios pueden manejar una gran variedad de protocolos como SOAP, XML/HTTP, HTTP RESTful, o CORBA, y pueden trabajar sobre transportes como HTTP, JMS o JBI.

Volver al índice

Spring Web Services

Se recomienda el uso de Spring Web Services

Es una extensión a Spring para facilitar la creación de servicios web Java. Spring-WS se basa en el concepto de "contrato primero", con el cual se define primero el contrato del servicio y luego se implementa, evitando atar al contrato como sucede en los casos en los cuales se genera el mismo a partir de las clases Java.

Volver al índice

Uso de especificaciones y estándares de servicios web

- **Área:** Integración
- **Tipo de pauta:** Directriz
- **Carácter de la pauta:** Recomendada

Código: LIBP-0320

Se deben seguir las recomendaciones sobre la conveniencia o no de utilizar cada una de las especificaciones y estándares relacionados con el mundo de ws recogidos en MADEJA, y que se relacionan a continuación

Pautas

Título	Carácter
SOAP v1.1, 1.2	Obligatoria
WSDL 2.0	Recomendada
JAX-RPC	No Recomendada
JAX-WS v2.0/2.1	Obligatoria
REST	No Recomendada
JAXB	Recomendada
WS Basic Profile v1.1	Obligatoria
WS-ReliableMessaging	Recomendada

SOAP v1.1, 1.2

Obligatorio para Servicios Web que se integren con PLATINA

Se recomienda el uso de los estándares SOAP (Simple Object Access Protocol) para el intercambio de información en entornos distribuidos, siendo obligatorio para aquellos servicios que vayan a integrarse con PLATINA.

Volver al índice

WSDL 2.0

Se recomienda el uso de WSDL 2.0

Los servicios web en sí son simples y versátiles, se trata de comunicaciones basadas en XML, descritos por una gramática basada en XML, llamada lenguaje descriptivo de servicios web (WSDL), el cual tiene interfaces abstractas de servicios, que consisten en mensajes expresados como un esquema XML.

Aunque no es un requerimiento, el formato elegido actualmente es SOAP sobre http. Esto significa que las interfaces de los servicios web están descritas en términos de los mensajes SOAP de entrada y salida, transmitidos sobre el protocolo HTTP.

Como modelo para la descripción de los servicios se recomienda el uso de WSDL 2.0 (Web Services Descripcion Language).

Volver al índice

JAX-RPC

No se recomienda el uso de JAX-RPC

Se trata de una API Java para RPC (Remote Procedure Call) basado en XML que permite la invocación de servicios web desarrollados en Java cuya descripción está basada en WSDL. JAX-RPC 2.0 fue renombrado a JAX-WS 2.0 (Java API for XML WebServices), quedando obsoleto a partir de Java EE 6.

Puede emplearse en versiones de java inferiores a la versión 1.5

Volver al índice

JAX-WS v2.0/2.1

Obligatorio para Java 1.5 y superiores

Se recomienda el uso de JAX-WS v2.0/2.1, siendo obligatorio a partir de la versión Java 1.5

Volver al índice

REST

No recomendado para servicios que se integren con PLATINA

Se trata de una arquitectura de servicios distribuidos que sigue una serie de requisitos:

- En REST se publican son ***recursos*** en vez de un conjunto de métodos u operaciones (RPC). Un recurso se puede considerar como una entidad que representa un concepto de negocio que puede ser accedido públicamente.
- Cada recurso posee un identificador único y global, que lo distingue de cualquier otro recurso, aunque ambos tuvieran exactamente los mismos datos.
- Cada recurso posee un estado interno, que no puede ser accedido directamente desde el exterior.

Volver al índice

JAXB

Recomendado para su uso con JAX-WS cuando el Servicio Web se implemente siguiendo el Modelo de Mensajes (SOA)

Se trata de una tecnología java que provee un API y una herramienta para ligar el esquema XML a una representación en código java. Con el uso de un esquema, se realiza una definición de los elementos que pueden estar contenidos en un documento XML. el esquema también es utilizado para la definición de la firma y la relación entre los elementos existentes.

Volver al índice

WS Basic Profile v1.1

Es obligatorio el uso de WS Basic Profile v1.1

Se trata de un conjunto de especificaciones para servicios web que promueven la interoperabilidad, como SOAP y WSDL.

Volver al índice

WS-ReliableMessaging

Se recomienda el uso de WS-ReliableMessaging

Describe un protocolo que permite el envío confiable de mensajes entre dos nodos frente a fallas en sistemas, componentes o conexión. El protocolo es independiente de la forma de transporte sin embargo se definió en esta especificación un enlace utilizando SOAP.

Se pueden especificar garantías de envío utilizando WS-Policy.

Volver al índice

Recursos

Área: Arquitectura » Integración

Código	Título	Tipo	Carácter
RECU-0100	Especificaciones y estándares de servicios web	Especificación	Recomendado

Área: Desarrollo » Seguridad » Servicios Web

Código	Título	Tipo	Carácter
RECU-0217	WS-Security	Especificación	Recomendado
RECU-0598	WS-Addressing	Especificación	Recomendado

Especificaciones y estándares de servicios web

- **Área:** Integración
- **Carácter del recurso:** Recomendado

Código: RECU-0100
Tipo de recurso: Especificación

Introducción

A continuación se incluye un listado con las principales especificaciones relacionadas con el mundo de los ws impulsadas por organizaciones reconocidas a nivel internacional como OASIS, W3C, WS-I y JCP:

Características

Estándar
SOAP v1.1, 1.2
WSDL 2.0
JAX-RPC
JAX-WS v2.0/2.1
REST
JAXB
WS Basic Profile v1.1
WS-Policy
WS-Security
WS-Reliable
WS-Addressing

Enlace oficial

Web Services Interoperability Organization (WS-I)

Enlaces externos

- Web Services Interoperability Organization (WS-I)
- Organization for the Advancement of Structured Information Standards (OASIS)
- World Wide Web Consortium (W3C)
- Java Community Process (JCP)

Pautas

Área: Arquitectura » Integración

Código	Título	Tipo	Carácter
LIBP-0320	Uso de especificaciones y estándares de servicios web	Directriz	Recomendada

Source URL: http://127.0.0.1/servicios/madeja/contenido/recurso/100

Plataforma de Interoperabilidad de la Junta de Andalucía

- **Área:** Integración
- **Carácter del recurso:** Recomendado

Código: RECU-0019
Tipo de recurso: Ficha

Descripción

Introducción

Las siglas PLATINA corresponden a la **PLAT**aforma de **IN**teroperabilidad de la Junta de **A**ndalucía.

El objetivo principal de PLATINA es conseguir una plataforma tecnológica que facilite e implante un modelo común de interoperabilidad en la Administración de la Junta de Andalucía y que permita servir como infraestructura de soporte común para facilitar la integración e interoperabilidad entre servicios de la Junta de Andalucía y de distintas Administraciones.

De forma más detallada, los objetivos y el alcance de PLATINA se pueden resumir en:

- Disponer de una plataforma tecnológica de interoperabilidad que permita la integración de sistemas y servicios dentro de la Junta de Andalucía y con otras Administraciones Públicas (procesos intra e inter-administraciones). Esto hará posible agilizar los intercambios de información, aumentando la eficiencia y eficacia de los servicios públicos y ahorrando así a la ciudadadanía retrasos y trámites innecesarios.
- Creación y aprovisionamiento de servicios sobre aplicaciones ya existentes y futuras.
- Desarrollo de una metodología y diseño de desarrollos orientados a arquitectura SOA, alineado con el Marco de Desarrollo de la Junta de Andalucía (MADEJA) para:
  - Creación y promoción de nuevos servicios
  - Consumos de servicios
  - Gobierno de servicios
- Difusión y comunicación

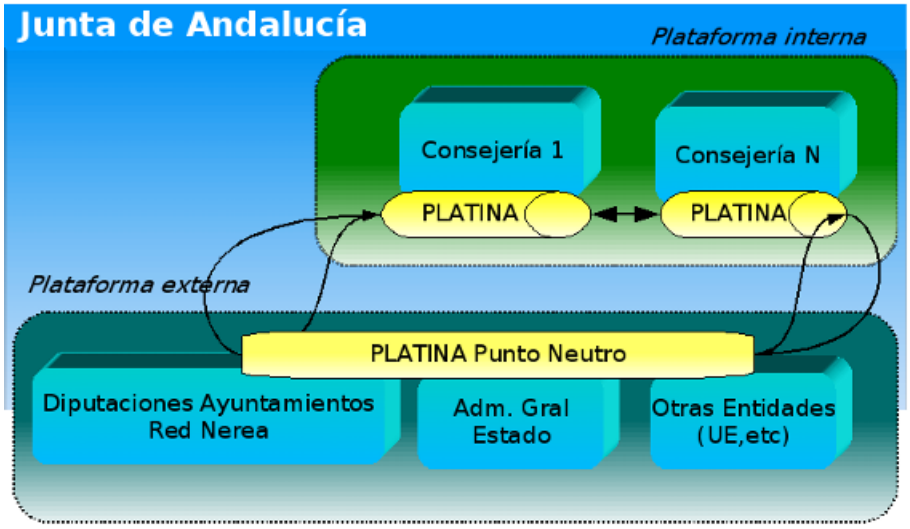
Con la consecución de los objetivos, se logrará:

- Favorecer el alineamiento de las Tecnologías de la Información de la Junta de Andalucía.
- Optimizar los costes de gestión e inversión en el desarrollo de software de la Junta de Andalucía, maximizando la reutilización.
- Independizar el software de los aplicativos respecto a las plataformas.
- Estandarizar la definición y despliegue de los servicios como vía para garantizar la interoperabilidad entre los sistemas.

La Plataforma de Interoperabilidad de la Junta de Andalucía (PLATINA) actuará como elemento mediador o enrutador de los servicios desplegados en distintas ubicaciones, facilitando en algunos casos una cierta inteligencia en las invocaciones mediante la orquestación de servicios. También servirá para la publicación y registro de los servicios existentes, de forma que se puedan clasificar y dar a conocer para su aprovechamiento y reutilización por cualquier sistema que los necesite.

Diseño

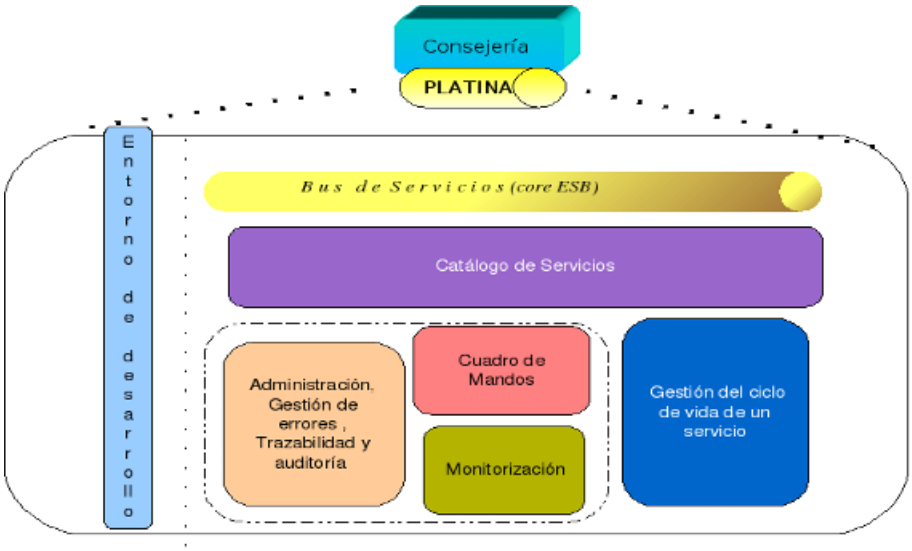
Arquitectura



PLATINA está formada por una arquitectura de nodos que se puede dividir en dos ámbitos bien diferenciados:

- **Plataforma Externa:** integración e intercambio de información telemática con otras Administraciones Públicas externas a la Junta de Andalucía (locales, estatales, europeas o internacionales). La Plataforma Externa está constituida por un nodo de PLATINA centralizado para comunicación con el exterior, al que se denomina Punto Neutro.
- **Plataforma Interna:**
  - Integración de servicios verticales o del ámbito de una única Consejería/Organismo. Se lleva a cabo mediante un nodo de PLATINA en cada Consejería y Organismo autónomo para el uso e integración de servicios verticales. Dicha infraestructura será compartida por todos los servicios que contenga un organismo.
  - Integración de servicios horizontales o que utilicen más de una Consejería/Organismo. Se requiere una arquitectura distribuida para la integración telemática de los sistemas de información de las Consejerías/Organismos de la Junta de Andalucía, con independencia de la tecnología que utilicen, tanto para prestar como para consumir servicios, así como para intercambiar información.
  - PLATINA ofrece también la posibilidad de implementar servicios en modalidad ASP (proveedor de aplicaciones) para aquellas entidades locales que no dispongan de recursos ni infraestructura para la implantación de soluciones de interoperabilidad

Diseño Funcional



Los subsistemas funcionales de diseño definidos para un nodo individual de PLATINA son:

- ESB
- Catálogo de Servicios
- Monitorización y Administración
  - Subsistema de Monitorización
    - Módulo de monitorización operacional
    - Monitorización de alertas e históricos
  - Subsistema de Administración
    - Módulo de configuración
    - Módulo de gestión de servicios
    - Módulo de alertas
  - Subsistema de auditoría y estadísticas
    - Módulo de Auditoría y Trazabilidad
    - Módulo de Estadísticas

Adicionalmente, la plataforma utiliza y ofrece protocolos, recursos y servicios de:

- Seguridad
- Alta Disponibilidad
- Gestión de errores

Mapa de Productos y Tecnologías

El diseño de PLATINA está basado en el empleo de componentes de Software Libre.

- **Infraestructura SOA (arquitectura orientada a servicios)**, con los siguientes componentes:
  - Bus de Integración JBI ("ESB: enterprise service bus"). Sirve como elemento mediador, enrutador y coordinador dentro de la infraestructura SOA

        - ESB: Open ESB + OpenMQ
            - JBI 1.0 RI
            - Sobre GlassFish (JEE 5)
            - Orquestador de servicios BPEL 2.0
            - Entorno de productividad NetBeans
            - Alta Disponibilidad
            - Transportes: HTTP(S), JMS, File, (S)FTP, eMail, etc.
            - Mensajería sincrona y asíncrona
    - Catálogo de Servicios: directorio para la publicación de servicios Web donde otras entidades puedan conocer la disponibilidad de los servicios prestados y disponibles por la plataforma.
        - Catálogo: freebXML. Implementación de la especificación de OASIS ebXML Registry/Repository
            - Registro y repositorio para almacenamiento de todo tipo de información sobre cada servicio (wsdl, manuales, ejemplos, apis, jar, etc.)
            - Consultas federadas
- **Gestión y Administración**
    - Herramienta de Administración, Auditoría, Estadísticas y Monitorización (ADAEMO)
    - BPEL Monitor
    - Hyperic HQ, Nagios
