



# **Guía Técnica de Publicación y Consumo de Servicios de Interoperabilidad**

## **Documento borrador**

**CTIC-IOP-P1-v.0.1**



**Abril de 2017**

## Índice de Contenido

1. MARCO NORMATIVO REFERENCIAL.....	3
2. OBJETIVO.....	3
3. ALCANCE DEL DOCUMENTO.....	4
4. NIVELES DE ACCESO A LA INFORMACIÓN INTEROPERABLE.....	4
5. PROVISIÓN DE SERVICIOS.....	4
6. TEMPORALIDAD.....	4
6.1. VERSIONAMIENTO DE LOS SERVICIOS REST.....	4
6.2. VERSIONAMIENTO DE LOS SERVICIOS SOAP.....	5
7. PUBLICACIÓN DE SERVICIOS.....	9
7.1. DOCUMENTO TÉCNICO DE SERVICIOS A CONSUMIR.....	9
7.2. VALIDACIÓN DE ESQUEMAS.....	9
7.2.1 VALIDACIÓN DE ESQUEMAS XML.....	9
7.2.2. VALIDACIÓN DE ESQUEMAS JSON.....	10
7.3. DESCUBRIMIENTO.....	12
7.3.1. DESCUBRIMIENTO EN SERVICIOS SOAP.....	12
7.3.2. DESCUBRIMIENTO EN SERVICIOS REST.....	13
7.4. INTERFAZ DE CONSUMO.....	13
7.5. GUÍA DE CÓDIGOS DE ESTADO HTTP.....	16
7.5.1 CÓDIGOS DE ESTADO PARA DIFERENTES TIPOS DE RESPUESTA DEL SERVICIO.....	16
7.5.2 CÓDIGOS DE ERROR.....	18
7.6. ENCABEZADOS.....	20
7.7. ESTADO DEL SERVICIO.....	21
8. ANEXOS.....	21
8. REFERENCIAS.....	22

# **GUÍA TÉCNICA DE PUBLICACIÓN Y CONSUMO DE SERVICIOS DE INTEROPERABILIDAD**

## **1. MARCO NORMATIVO REFERENCIAL**

El Parágrafo II del Artículo 103 de la Constitución Política del Estado, establece que el Estado asumirá como política la implementación de estrategias para incorporar el conocimiento y aplicación de nuevas tecnologías de información y comunicación.

Asimismo, el Parágrafo I del Artículo 85 de la Ley N° 031, de 19 de julio de 2010, Marco de Autonomías y Descentralización "Andrés Báñez", determina que dentro la competencias exclusivas del nivel central del Estado, se encuentra formular y aprobar el régimen general y las políticas de comunicaciones y telecomunicaciones del país, incluyendo el acceso al internet y demás Tecnologías de Información y Comunicaciones - TIC.

El Artículo 72 de la Ley N° 164, de 8 de agosto de 2011, de Telecomunicaciones, Tecnologías de Información y Comunicación, señala que las entidades públicas deberán adoptar todas las medidas necesarias para garantizar el máximo aprovechamiento de las tecnologías de información, de manera prioritaria, haciendo énfasis en el área de gestión gubernamental, como mecanismo para atender la demanda social, facilitar el acceso y uso intensivo a nivel interno de cada unidad gubernamental, entre entidades gubernamentales, entre las ciudadanas y ciudadanos con las entidades gubernamentales, concordante con el Decreto Supremo N° 28168, de 17 de mayo de 2005, de Acceso a la Información, que dispone el derecho de acceso a la información a todas las personas, el cual sólo podrá ser negado de manera excepcional y motivada, únicamente respecto a aquella información que con anterioridad a la petición y de conformidad a leyes vigentes se encuentre clasificada como secreta, reservada o confidencial.

El Artículo 18 del Decreto Supremo N° 1793, de 13 de noviembre de 2013, Reglamento de la Ley N° 164, de 8 de abril de 2011, de Telecomunicaciones, Tecnologías de Información y Comunicación, señala que el Plan de Implementación del Gobierno Electrónico, deberá considerar minimamente los siguientes lineamientos: d) Proponer mecanismos para lograr eficiencia en el uso de los recursos tecnológicos de las entidades públicas, además de la interoperabilidad de los sistemas de información y de servicios gubernamentales desarrollados por cada una de ellas, a través de la aplicación y uso de estándares abiertos.

Finalmente, el Artículo 19 del Decreto Supremo N° 2514 de 9 de Septiembre de 2015, instituye a la AGETIC a coordinar con las entidades del sector público, la implementación de servicios de interoperabilidad de Gobierno Electrónico, así como los datos e información que deben estar disponibles, autorizando a las entidades públicas proporcionar a la AGETIC los datos e información que hubieran producido, recolectado o generado, por medios electrónicos o mecanismos de interoperabilidad, que ésta solicite mediante nota formal de su MAE, en el marco de la política general de Gobierno Electrónico, simplificación de trámites, transparencia, participación y control social y tecnologías de la información y comunicación; siendo el ente rector de Gobierno Electrónico, quien determinará la política general y normativa específica de interoperabilidad e intercambio de información y datos entre las entidades del sector público.

## **2. OBJETIVO**

El presente documento tiene el objeto de establecer la guía técnica que permita estandarizar la publicación y consumo de servicios web para la interoperabilidad en las entidades del sector público.

## **3. ALCANCE DEL DOCUMENTO**

Este documento es una guía para estandarizar la publicación y consumo de servicios, y no para imponer o limitar el modo de compartir información entre las entidades. No obstante al ser un documento consensuado entre las mismas, éstas serán las reglas que se sugieren seguir para tener una buena interoperabilidad entre entidades.

## **4. NIVELES DE ACCESO A LA INFORMACIÓN INTEROPERABLE**

El Artículo 72 de la Ley N° 164, de 8 de agosto de 2011, de Telecomunicaciones, Tecnologías de Información y Comunicación, señala que las entidades públicas deberán adoptar todas las medidas necesarias para garantizar el máximo aprovechamiento de las tecnologías de información, de manera prioritaria, haciendo énfasis en el área de gestión gubernamental, como mecanismo para atender la demanda social, facilitar el acceso y uso intensivo a nivel interno de cada unidad gubernamental, entre entidades gubernamentales, entre las ciudadanas y ciudadanos con las entidades gubernamentales. Concordante con el Decreto Supremo N° 28168, de 17 de mayo de 2005, de Acceso a la Información, que dispone el derecho de acceso a la información a todas las personas, el cual sólo podrá ser negado de manera excepcional y motivada, únicamente respecto a aquella información que con anterioridad a la petición y de conformidad a leyes vigentes se encuentre clasificada como secreta, reservada o confidencial por parte de las entidades.

En ese sentido, la información que se publicará por medio de los servicios de interoperabilidad de las entidades deben estar acorde a las pautas expuestas anteriormente, pudiendo considerarse el intercambio de la información considerada pública y/o a través de convenios.

## **5. PROVISIÓN DE SERVICIOS**

El proveedor del servicio web debe adjuntar a su documentación el tiempo de duración del servicio y otras especificaciones técnicas que sirvan para consulta de los clientes que consuman el servicio.

## **6. TEMPORALIDAD**

### **6.1. VERSIONAMIENTO DE LOS SERVICIOS REST**

Con el propósito de asegurar la continuidad de la provisión del servicio, la entidad proveedora del servicio deberá asociar un número de versión a la interfaz de consumo en todos los servicios. Para el caso de una actualización de la interfaz, el proveedor debe mantener la versión hasta que ningún cliente este consumiendo el servicio. El productor deberá incrementar el número de versión, adicionalmente deberá informar a los consumidores de la existencia de una nueva versión incluyendo toda la información técnica detallada sobre los cambios realizados.

#### **Criterio para versionar servicios de interoperabilidad**

Se define un número de versión MAYOR.MENOR.REVISIÓN (Ejemplo: v1.2.1):

- Donde el número de versión MAYOR es incrementado cuando se realizan cambios del servicio

parcialmente o totalmente incompatibles con la versión anterior.

- Donde el número de versión MENOR es incrementado cuando se agrega nueva funcionalidad compatible con versiones anteriores.
- Donde el número de versión REVISIÓN es incrementada cuando se realiza correcciones de errores y son compatibles con versiones anteriores.

El proveedor debe especificar un número de versión del servicio de orden MAYOR en las URLs de su servicio.

Ejemplo:

```
GET /v1/tramites
GET /v2/tramites
...
GET /v(n)/tramites
```

Para las versiones MENORES y de REVISIÓN, el consumidor (opcionalmente) puede especificar en el encabezado de su solicitud la versión que quiere consumir. El servicio del proveedor en caso de no recibir la especificación de la versión a consumir debe tomar por defecto la ultima versión publicada<sup>1</sup>.

Ejemplo:

```
URL: /api/v2/recurso
Method: GET
Headers:
    api-version: 2.4.3
```

## 6.2. VERSIONAMIENTO DE LOS SERVICIOS SOAP

Después de la implementación inicial y luego de haber transcurrido un tiempo de disponibilidad y uso, los servicios (y los extremos que exponen) pueden necesitar ser cambiados debido a diversas razones, como cambios en las necesidades comerciales, requisitos de tecnología de la información o para resolver otros problemas. Cada cambio produce una nueva versión del servicio.

Los cambios en los servicios que se pueden requerir pueden clasificarse en cuatro categorías:

- Cambios del contrato: por ejemplo, se podría agregar una operación o se podría agregar o cambiar un elemento de datos en un mensaje.
- Cambios de la dirección: por ejemplo, se mueve un servicio a una ubicación diferente donde los extremos tienen nuevas direcciones.
- Cambios del enlace: por ejemplo, un mecanismo de seguridad cambia o lo hace su configuración.
- Cambios de implementación: por ejemplo, cuando cambia una implementación de método interno.

Algunos de estos cambios se denominan "con interrupción" y otros son denominados "sin interrupción". Un cambio es *sin interrupción* si todos los mensajes que se habrían procesado correctamente en la versión anterior se procesan correctamente en la nueva versión. Cualquier cambio que no cumpla ese criterio es un cambio *con interrupción*.

---

<sup>1</sup> Semantic Versioning 2.0.0 <http://semver.org/lang/es/>  
REST y el versionado de servicios <https://www.adictosaltrabajo.com/tutoriales/rest-y-el-versionado-de-servicios/>

En los casos en los que no se prevé un cambio con interrupción y no puede evitarse, una aplicación puede decidir si omitir este principio y requerir que los clientes se re-compilen y se implementen de nuevo con una nueva versión del servicio.

La recomendación general para el control de versiones de los contratos de datos es que cuando se necesite un control estricto de las versiones **los contratos de datos se traten como inmutables** y crear otros nuevos cuando se requieran cambios. Se necesita crear una nueva clase para cada nuevo contrato de datos, por lo que necesita que un mecanismo evite tener que tomar código existente, que se escribió referido a la clase del contrato de datos antiguo y volverlo a escribir, referido a la nueva clase del contrato de datos.

Dicho mecanismo se utiliza en interfaces para definir los miembros de cada contrato de datos y escribir el código de implementación interno, referido a las interfaces en lugar de referirse a las clases del contrato de datos que implementan las interfaces. Como ejemplo, si un método tiene el siguiente contrato:

metodoV1(param1, param2)

Entonces, cuando el contrato se modifica (cambian los parámetros), el nuevo método es:

metodoV2(param1, param2, param3, param4)

## Implementación de versiones del servicio

1°. Cree la configuración de servicio de enrutamiento básica especificando el extremo de servicio expuesto por el servicio. En el siguiente ejemplo, se define un extremo de servicio único que se utilizará para recibir mensajes. También se definen los extremos del cliente que se utilizarán para enviar mensajes a los servicios roundingCalc (v1) y regularCalc (v2).

```
<services>
  <service behaviorConfiguration="routingConfiguration"
    name="System.ServiceModel.Routing.RoutingService">
    <host>
      <baseAddresses>
        <add baseAddress="http://localhost/routingservice/router" />
      </baseAddresses>
    </host>
    <!--Set up the inbound endpoint for the Routing Service-->
    <endpoint address="calculator"
      binding="wsHttpBinding"
      name="routerEndpoint"
      contract="System.ServiceModel.Routing.IRequestReplyRouter" />
  </service>
</services>
<client><!--set up the destination endpoints-->
  <endpoint name="regularCalcEndpoint"
    address="net.tcp://localhost:9090/servicemodelsamples/service/"
    binding="netTcpBinding"
    contract="*" />
  <endpoint name="roundingCalcEndpoint"
    address="http://localhost:8080/servicemodelsamples/service/"
    binding="wsHttpBinding"
```

```

        contract="*" />
    </client>

```

2°. Defina los filtros usados para enrutar mensajes a los extremos del destino. En este ejemplo, el filtro XPath se usa para detectar el valor del encabezado personalizado "CalcVer" para determinar a qué versión debe enrutarse el mensaje. También se utiliza un filtro XPath para detectar mensajes que no contienen el encabezado "CalcVer". En el siguiente ejemplo, se definen los filtros necesarios y la tabla de espacio de nombres.

```

<!-- use the namespace table element to define a prefix for our custom namespace-->
<namespaceTable>
    <add prefix="custom" namespace="http://my.custom.namespace/" />
</namespaceTable>
<filters>
    <!--define the different message filters-->
    <!--define an xpath message filter to look for the custom header containing a value of 2-->
    <filter name="XPathFilterRegular" filterType="XPath" filterData="sm:header()/custom:CalcVer
= '2'"/>
    <!--define an xpath message filter to look for the custom header containing a value of 1-->
    <filter name="XPathFilterRounding" filterType="XPath" filterData="sm:header()/custom:CalcVer
= '1'"/>
    <!--define an xpath message filter to look for messages that do not contain the custom
header-->
    <filter name="XPathFilterNoHeader" filterType="XPath"
filterData="count(sm:header()/custom:CalcVer)=0"/>
</filters>

```

3°. Defina la tabla de filtro, que asocia cada filtro a un extremo del cliente. Si el mensaje contiene el encabezado "CalcVer" con un valor de 1, se enviará al servicio de regularCalc. Si el encabezado contiene un valor de 2, se enviará al servicio de roundingCalc. Si no hay ningún encabezado, el mensaje se enrutará a regularCalc.

El procedimiento siguiente define la tabla de filtros y agrega los filtros definidos anteriormente.

```

<filterTables>
    <filterTable name="filterTable1">
        <!--add the filters to the message filter table-->
        <!--look for the custom header = 1, and if we find it,
            send the message to the rounding calc endpoint-->
        <add filterName="XPathFilterRounding" endpointName="roundingCalcEndpoint"/>
        <!--look for the custom header = 2, and if we find it,
            send the message to the rounding calc endpoint-->
        <add filterName="XPathFilterRegular" endpointName="regularCalcEndpoint"/>
        <!--look for the absence of the custom header, and if
            it is not present, assume the vl endpoint-->
        <add filterName="XPathFilterNoHeader" endpointName="roundingCalcEndpoint"/>
    </filterTable>
</filterTables>

```

4°. Para evaluar los mensajes entrantes con respecto a los filtros incluidos en la tabla de filtros, debe asociar esta a los extremos de servicio mediante el comportamiento de enrutamiento. En el siguiente ejemplo, se muestra cómo asociar "filterTable1" a los extremos de servicio:

```

<behaviors>

```

```

<!--default routing service behavior definition-->
<serviceBehaviors>
  <behavior name="routingConfiguration">
    <routing filterTableName="filterTable1" />
  </behavior>
</serviceBehaviors>
</behaviors>

```

## Ejemplo

A continuación, se muestra una lista completa del archivo de configuración.

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <services>
      <service behaviorConfiguration="routingConfiguration"
        name="System.ServiceModel.Routing.RoutingService">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost/routingservice/router" />
          </baseAddresses>
        </host>
        <!--Set up the inbound endpoint for the Routing Service-->
        <endpoint address="calculator"
          binding="wsHttpBinding"
          name="routerEndpoint"
          contract="System.ServiceModel.Routing.IRequestReplyRouter" />
      </service>
    </services>
    <behaviors>
      <!--default routing service behavior definition-->
      <serviceBehaviors>
        <behavior name="routingConfiguration">
          <routing filterTableName="filterTable1" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <client><!--set up the destination endpoints-->
      <endpoint name="regularCalcEndpoint"
        address="net.tcp://localhost:9090/servicemodelsamples/service/"
        binding="netTcpBinding"
        contract="*" />
      <endpoint name="roundingCalcEndpoint"
        address="http://localhost:8080/servicemodelsamples/service/"
        binding="wsHttpBinding"
        contract="*" />
    </client>
    <routing>
      <!-- use the namespace table element to define a prefix for our custom namespace-->
      <namespaceTable>
        <add prefix="custom" namespace="http://my.custom.namespace/" />
      </namespaceTable>
    </routing>
  </system.serviceModel>
</configuration>

```



```

<filters>
  <!--define the different message filters-->
  <!--define an xpath message filter to look for the
    custom header containing a value of 2-->
  <filter name="XPathFilterRegular" filterType="XPath"
    filterData="sm:header()/custom:CalcVer = '2'"/>
  <!--define an xpath message filter to look for the
    custom header containing a value of 1-->
  <filter name="XPathFilterRounding" filterType="XPath"
    filterData="sm:header()/custom:CalcVer = '1'"/>
  <!--define an xpath message filter to look for
    messages that do not contain the custom header-->
  <filter name="XPathFilterNoHeader" filterType="XPath"
    filterData="count(sm:header()/custom:CalcVer)=0"/>
</filters>
<filterTables>
  <filterTable name="filterTable1">
    <!--add the filters to the message filter table-->
    <!--look for the custom header = 1, and if we find it,
      send the message to the rounding calc endpoint-->
    <add filterName="XPathFilterRounding" endpointName="roundingCalcEndpoint"/>
    <!--look for the custom header = 2, and if we find it,
      send the message to the rounding calc endpoint-->
    <add filterName="XPathFilterRegular" endpointName="regularCalcEndpoint"/>
    <!--look for the absence of the custom header, and if
      it is not present, assume the v1 endpoint-->
    <add filterName="XPathFilterNoHeader" endpointName="roundingCalcEndpoint"/>
  </filterTable>
</filterTables>
</routing>
</system.serviceModel>
</configuration>

```

## 7. PUBLICACIÓN DE SERVICIOS

### 7.1. DOCUMENTO TÉCNICO DE SERVICIOS A CONSUMIR

Es recomendable adjuntar al convenio de interoperabilidad un documento que contenga todos los aspectos técnicos de los servicios a consumir para que los encargados de las Tecnologías de la Información y Comunicación (áreas de sistemas o informáticas) puedan utilizar el mismo para la implementación de sus servicios bajo los parámetros citados en dicho documento. Este documento debe contener el listado de servicios solicitados, parámetros que se utilizaran para cada solicitud, etc.

### 7.2. VALIDACIÓN DE ESQUEMAS

#### 7.2.1 VALIDACIÓN DE ESQUEMAS XML

Algunos esquemas XML requieren que especifique el espacio de nombres XML que desea validar, otros no. Si se proporciona un valor en el parámetro URI de espacio de nombre cuando no se requiere nada, se relacionarán las advertencias o errores informados con el hecho de no encontrar la información correcta en el

esquema XML y no se identificarán si el documento XML es válido.

Por ejemplo, si intenta validar un documento XML que cumple con el Comité Federal de Datos Geográficos (FGDC) Estándar de contenidos para metadatos geoespaciales digitales (CSDGM) utilizando el esquema XML <http://www.fgdc.gov/schemas/metadata/fgdc-std-001-1998.xsd>, no debe proporcionar un valor en el parámetro Namespace URI.

### Ejemplo de código

Validar un archivo XML de metadatos ISO 19139 utilizando un Esquema XML:

- Valida un archivo XML independiente que contiene metadatos con formato ISO 19139 utilizando el Esquema XML ISO 19139 online. Estos Esquemas XML requieren que especifique el namespace XML que desea validar.

```
import arcpy
from arcpy import env
env.workspace = "C:/data"
#set local variables
schema = "http://www.isotc211.org/schemas/2005/gmd/metadataEntity.xsd"
namespace = "http://www.isotc211.org/2005/gmd"
arcpy.XMLSchemaValidator_conversion("metadata_19139.xml", schema, namespace)
```

Validar un archivo XML de metadatos FGDC utilizando un Esquema XML:

- Valida un archivo XML independiente que contiene metadatos con formato CSDGM FGDC utilizando una copia local de los archivos de Esquema XML FGDC. Estos se pueden descargar desde el sitio Web de estándares de metadatos FGDC. Estos Esquemas XML no requieren que especifique el namespace XML que desea validar.

```
import arcpy
from arcpy import env
env.workspace = "C:/data"#set local variables
schema = "c:/metadata/FGDCxsd/non-annotated/fgdc-std-001-1998.xsd"
arcpy.XMLSchemaValidator_conversion("metadata_fgdc.xml", schema, "#")
```

### Validación de esquemas XML (XSD) con XmlSchemaCollection

Puede utilizar XmlSchemaCollection para validar un documento XML con esquemas del lenguaje de definición de esquemas XML (XSD). XmlSchemaCollection mejora el rendimiento al almacenar esquemas en la colección para que no se carguen en memoria cada vez que se produce la validación. Si el esquema está en la colección de esquemas, el atributo schemaLocation se utiliza para buscarlo en dicha colección.

En el ejemplo siguiente se muestra el elemento raíz de un archivo de datos.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:bookstore-schema"
  elementFormDefault="qualified"
  targetNamespace="urn:bookstore-schema">
```

Para este ejemplo, el valor del atributo targetNamespace es urn:bookstore-schema, que es el mismo espacio de

nombres que se ha utilizado al agregar el esquema a XmlSchemaCollection.

El siguiente código de ejemplo agrega un esquema XML a XmlSchemaCollection.

```
XmlSchemaCollection xsc = new XmlSchemaCollection();// XML Schema.  
xsc.Add("urn:bookstore-schema", schema);  
reader = new XmlTextReader (filename);  
vreader = new XmlValidatingReader (reader);  
vreader.Schemas.Add(xsc);
```

Generalmente, se utiliza el atributo targetNamespace al agregar la propiedad namespaceURI en el método Add para XmlSchemaCollection. Puede especificar una referencia nula antes de agregar el esquema a XmlSchemaCollection. Se debería utilizar una cadena vacía ("") para los esquemas que no tengan un espacio de nombres. XmlSchemaCollection solo puede tener un esquema sin espacio de nombres.

### 7.2.2. VALIDACIÓN DE ESQUEMAS JSON

JSON Schema es a JSON lo que XSD a XML, es decir un formato JSON para describir datos en JSON. Es una especificación para el formato JSON con base para definir la estructura de los datos JSON. Fue escrito en virtud del proyecto IETF que expiró en 2011.

#### Características del Esquema JSON

Describe el formato de los datos existentes, Claro, documentación legible por máquina, útil para realizar pruebas automáticas, validación de los datos del cliente antes de procesarlos.

#### Bibliotecas de validación de JSON esquema

Hay varios validadores actualmente disponibles para diferentes lenguajes de programación. Actualmente el validador más completo y compatible con JSON Esquema disponible es JSV. <http://json-schema.org/implementations.html>

Lenguajes	Bibliotecas
C	WJElement (LGPLv3)
java	JSON-esquema-validador (LGPLv3)
.NET	Json.NET(MIT)
ActionScript 3	Frigga (MIT)
Haskell	Aeson-esquema (MIT)
Phyton	Jsonschema
Rubí	autoparse (ASL 2.0); rubí jsonschema (MIT)

Lenguajes	Bibliotecas
PHP	php-JSON-esquema (MIT). JSON-esquema (Berkeley)
JavaScript	Ordenada (BSD); JSV; JSON-esquema; Matic (MIT); dojo; Perseverar (BSD modificada o AFL 2.0); schema.js

## Ejemplo de esquema

A continuación se realiza un esquema JSON básico, que cubre una descripción clásica catálogo de productos

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Product",
  "description": "A product from Acme's catalog",
  "type": "object",

  "properties": {
    "id": {
      "description": "The unique identifier for a product",
      "type": "integer"
    },

    "name": {
      "description": "Name of the product",
      "type": "string"
    },

    "price": {
      "type": "number",
      "minimum": 0,
      "exclusiveMinimum": true
    }
  },

  "required": ["id", "name", "price"]
}
```

En la url <http://json-schema.org/latest/json-schema-validation.html#rfc.section.5> esta la lista completa de palabras clave que se pueden utilizar en la definición de un esquema JSON.

El esquema anterior puede ser utilizado para probar la validez del siguiente código JSON<sup>2</sup>:

```
[
  {
    "id": 2,
    "name": "An ice sculpture",
```

<sup>2</sup> JSON Schema – Software  
<http://json-schema.org/implementations.html>

```

        "price": 12.50,
    },
    {
        "id": 3,
        "name": "A blue mouse",
        "price": 25.50,
    }
]

```

## 7.3. DESCUBRIMIENTO.

### 7.3.1. DESCUBRIMIENTO EN SERVICIOS SOAP

El descubrimiento de servicios Web XML es el proceso consistente en localizar, o descubrir, uno o varios documentos relacionados que describen un servicio Web XML determinado mediante Lenguaje de descripción de servicios Web (WSDL). A través del proceso de descubrimiento, los clientes de servicios Web XML conocen la existencia de un servicio Web XML y dónde encontrar el documento de descripción del mismo.

Un archivo *.disco* publicado, es un documento XML donde se incluyen vínculos a otros recursos que describen el servicio Web XML, permite descubrir mediante programación un servicio Web XML. A continuación se muestra un ejemplo de la estructura de un documento de descubrimiento:

```

<?xml version="1.0" encoding="utf-8" ?>
<discovery xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <contractRef ref="http://www.contoso.com/Counter.asmx?wsdl"
    docRef="http://www.contoso.com/Counter.asmx"
    xmlns="http://schemas.xmlsoap.org/disco/scl/" />
  <soap address="http://www.contoso.com/Counter.asmx"
    xmlns:q1="http://tempuri.org/"
    binding="q1:CounterSoap"
    xmlns="http://schemas.xmlsoap.org/disco/soap/" />
</discovery>

```

**Nota.** - El documento de descubrimiento es un contenedor de elementos que suelen incluir vínculos (direcciones URL) a recursos que proporcionan información de descubrimiento para un servicio Web XML. Si las direcciones URL son relativas, se supone que lo son con respecto a la ubicación del documento de descubrimiento.

Sin embargo, un sitio Web que implementa un servicio Web XML no necesariamente tiene que permitir el descubrimiento. Otro sitio podría encargarse de la descripción del servicio; por ejemplo, un directorio de servicios Web XML. También existe de la posibilidad de que no haya un medio público para buscar el servicio; por ejemplo, cuando se crea para uso privado<sup>3</sup>.

### 7.3.2. DESCUBRIMIENTO EN SERVICIOS REST

Para utilizar la API de REST, siga uno o varios de estos pasos:

Para comprobar el estado del descubrimiento actual, utilice el recurso de estado del descubrimiento y

<sup>3</sup> Descubrimiento de servicios Web XML  
[https://msdn.microsoft.com/es-es/library/aa735703\(v=vs.71\).aspx](https://msdn.microsoft.com/es-es/library/aa735703(v=vs.71).aspx)

especifique el formato XML o JSON para los datos de la salida. En este ejemplo se comprueba el estado del descubrimiento utilizando el formato JSON:

```
http://example.com/rest/discovery/status?feed=json
```

Para recuperar una lista de los perfiles de descubrimiento definidos, utilice el recurso del servicio del perfil de descubrimiento y especifique el formato XML o JSON para los datos de salida. En este ejemplo se muestran los perfiles de descubrimiento que utilizan el formato XML:

```
http://example.com/rest/discovery/profiles?feed=xml
```

Para recuperar los detalles de un perfil de descubrimiento definido, utilice el recurso del perfil de descubrimiento y especifique el formato XML o JSON para los datos de salida. Así, se recuperan los detalles del perfil Descubrimiento de nivel 3 utilizando el formato JSON:

```
http://example.com/rest/discovery/profile/Level%203%20Discovery?feed=json
```

Para recuperar una lista de ámbitos de descubrimiento definidos, utilice el recurso del servicio del ámbito de descubrimiento y especifique el formato XML o JSON para los datos de salida. En este ejemplo, se muestran los ámbitos de descubrimiento utilizando el formato XML:

```
http://example.com/rest/discovery/scopes?feed=xml
```

Para recuperar los detalles de un ámbito de descubrimiento definido, utilice un recurso del ámbito de descubrimiento y especifique el formato XML o JSON para los datos de salida. Así, se recuperarán los detalles del ámbito scope1 utilizando el formato JSON<sup>4</sup>:

```
http://example.com/rest/discovery/scope/scope1?feed=json
```

## 7.4. INTERFAZ DE CONSUMO

Puntos a tomar en cuenta en URLs

### Nombres

Se deben usar nombres y no verbos, p.e.:

GET /tramites no usar GET /obtenerTramites

### Plurales

Se deben usar nombres en plurales, no nombres singulares. Considerar los tipos de recursos de tipo colección y de tipo instancia, p.e.:

Recurso de tipo colección: GET /tramites

```
{
  "tramites": [
    {
      "idTramite": 789,
      "idMensaje": 963,

```

---

<sup>4</sup> Ventajas e inconvenientes de API REST para el desarrollo  
<https://desarrolloweb.com/articulos/ventajas-inconvenientes-api-rest-desarrollo.html>

```

        "descripcion":"Trámite de renovación de licencia de conducir"
      },
      {
        "idTramite":987,
        "idMensaje":369,
        "descripcion":"Trámite de renovación de cédula de indentidad"
      }
    ]
  }
}

```

Recurso de tipo instancia: GET /tramites/1234

```

{
  "idTramite":789,
  "idMensaje":963,
  "descripcion":"Trámite de renovación de licencia de conducir"
}

```

### Estructura Jerárquica

Se debe aprovechar la naturaleza jerárquica de la URL, por ejemplo un trámite que tiene asociado una instancia de trámite:

```
GET /tramites/1234/instancias/4321
```

### Versionamiento

Se debe hacer versionamiento de forma obligada en la URL, por ejemplo:

```
GET /v1/tramites
```

Se podrá tener hasta dos versiones al mismo tiempo.

### Nombrado Consistente

Se debe usar camel case para el nombrado de los recursos, atributos y parámetros, por ejemplo:

```

GET /instanciasTramite

GET /modosPago?idEntidad=2345

POST /tramites {"idEntidad":"2345"}

```

### Operaciones al estilo CRUD

Se deben usar los verbos HTTP para las operaciones CRUD (Create/Read/Update/Delete), como se muestra a continuación:

Verbo HTTP	Colección: /recurso	Instancia: /recurso/:id
GET	Leer una lista de recursos. 200 OK.	Lee los detalles de una instancia de recurso. 200 OK.
POST	Crear un nuevo recurso. 201 Created. Devolver o no el recurso creado es decisión de la implementación, solo	

Verbo HTTP	Colección: /recurso	Instancia: /recurso/:id
	mantener la consistencia.	
PUT		Actualización completa. 200 OK. / Crear una instancia específica. 201 Created.
PATCH		Actualización parcial. 200 OK.
DELETE		Recurso eliminado. 204 OK.

## Cadenas de Consulta

### Búsqueda

Se debe usar la palabra clave "busqueda" para ejecutar una búsqueda en un recurso específico, por ejemplo:

```
GET    /tramites/busqueda?mensaje=*seguro*
```

### Filtros

Se debe usar "?" para filtrar los recursos y "&" para añadir más filtros, por ejemplo:

```
GET    /tramites?idEntidad=2345&estado=pendiente
```

### Paginación

Se debe definir la paginación de forma obligatoria, la respuesta deberá contener las cabeceras Content-Range y Accept-Range de la siguiente manera:

```
Content-Range: offset-limit/count
```

```
Accept-Range: resource max
```

donde:

- intervalo(offset): Índice del primer elemento devuelto.
- limite (limit): Cantidad de datos devueltos (-1 indica que se requieren todos los datos, sin embargo, este valor se debe aplicar solamente para listados que no devuelvan una gran cantidad de datos).
- count: El número total de elementos que contiene la colección.
- resource: El recurso solicitado.
- max: Número máximo de elementos en una petición.

Por ejemplo:

```
GET    /tramites?limite=10&intervalo=51
```

```
206 Partial Content
```

```
Content-Range: 51-60/100
```

```
Accept-Range: tramite 20
```



## Vínculos de Navegación (HATEOAS)

Se debe enviar en la respuesta la cabecera Link, en el ejemplo anterior:

Link:

```
</v1/tramites?limite=10&intervalo=1>; rel="first",  
</v1/tramites?limite=10&intervalo=41>; rel="prev",  
</v1/tramites?limite=10&intervalo=61>; rel="next",  
</v1/tramites?limite=10&intervalo=91>; rel="last"
```

## Respuestas Parciales

Para poder seleccionar la información que se desea recuperar y optimizar el ancho de banda, se deberá utilizar el parámetro "campos", p.e.:

```
GET    /tramites/1234?campos=descripcion,idEntidad,mensaje(mensaje)  
  
200 OK  
  
{  
  "idTramite": "1234",  
  "descripcion": "Trámite de renovación de licencia de conducir",  
  "idEntidad": "2345",  
  "mensaje": { "mensaje": "¿Está seguro de realizar el pago de 20 Bs?" }  
}
```

## Orden

Para poder ordenar los recursos se deberá usar el parámetro "orden", por defecto los recursos están ordenados de manera ascendente, p.e.:

```
GET    /tramites?orden=estado,descripcion
```

Para ordenar los recursos de forma descendente usar el parámetro "-", para ordenar de manera ascendente utilizar el parámetro "+", por defecto el orden es ascendente, p.e.:

```
GET    /tramites?orden=estado,-descripcion
```

## 7.5. GUÍA DE CÓDIGOS DE ESTADO HTTP

Con el fin de crear un estándar en los códigos de respuesta de los servicios REST se sugiere utilizar los siguientes:

### 7.5.1 CÓDIGOS DE ESTADO PARA DIFERENTES TIPOS DE RESPUESTA DEL SERVICIO

#### Success

200	OK	Código básico de éxito. Funciona para los casos generales. Usado especialmente en la respuesta exitosa de GET o el contenido actualizado de PUT/PATCH.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>	

	CURL -i -H "Accept: application/json" https://api.fakecompany.com/v1/tramites/1234  <b>Respuesta:</b>  <pre>HTTP/1.1 200 {   "idTramite": "1234",   "descripcion": "Trámite de renovación de licencia de conducir",   "idEntidad": "2345",   "mensaje": "¿Está seguro de realizar el pago de 20 Bs?" }</pre>	
201	Created	Indica que el recurso fue creado. Típicamente es la respuesta a solicitudes PUT o POST.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -X POST -H "Accept: application/json" -d '{"descripcion": "Trámite de renovación de licencia de conducir", "idEntidad": "2345", "mensaje": "¿Está seguro de realizar el pago de 20 Bs?"}' https://api.fakecompany.com/v1/tramites/1234  <b>Respuesta:</b>  <pre>HTTP/1.1 201 {   "mensaje": "El tramite fue creado",   "tramite": {     "idTramite": "1234",     "descripcion": "Trámite de renovación de licencia de conducir",     "idEntidad": "2345",     "mensaje": {"mensaje": "¿Está seguro de realizar el pago de 20 Bs?"}   } }</pre>	
202	Accepted	Indica que la solicitud ha sido aceptada para procesamiento. Típicamente es la respuesta para una llamada a procesamiento asíncrono.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -X PUT -H "Accept: application/json" -d '{"idTramite": "1234", "estado": "Transferir a otro departamento"}' https://api.fakecompany.com/v1/tramites/  <b>Respuesta:</b>  <pre>HTTP/1.1 202 {   "mensaje": "El tramite 1234 esta en proceso para su derivacion" }</pre>	
204	No Content	La solicitud ha tenido éxito, pero no hay nada que mostrar. Frecuentemente enviado luego de DELETE exitoso.
	<b>Ejemplo de Consumo</b>	

	<b>Solicitud:</b>  CURL -i -X DELETE -H "Accept: application/json, Token: hgtfdxcxfhhgfgbb..." -d '{"idTramite":"1234"}' https://api.fakecompany.com/v1/tramites/	
	<b>Respuesta:</b>  HTTP/1.1 204	
206	Partial Content	Recurso retornado está incompleto. Típicamente usado con recursos paginados.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -H "Accept: application/json" -d '{"page": "3", "limit": 50}' https://api.fakecompany.com/v1/tramites/	
	<b>Respuesta:</b>  HTTP/1.1 206 <pre> {   "metadatos": {     "total": 1520,     "page": 3,     "limit": 50   },   "tramites": [     {       "idTramite": "1234",       "descripcion": "Trámite de renovación de licencia de conducir",       "idEntidad": "2345",       "mensaje": "¿Está seguro de realizar el pago de 20 Bs?"     },     {       "idTramite": "1235",       "descripcion": "Trámite de nueva licencia de conducir",       "idEntidad": "2666",       "mensaje": "¿Está seguro de realizar el pago de 20 Bs?"     },     .     .     .   ] }</pre>	

## 7.5.2 CÓDIGOS DE ERROR

### Client Error

400	Bad Request	Error general para una solicitud que no puede ser procesada (el cliente no debe repetir la solicitud sin modificarla).
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>	

	CURL -i -H "Accept: application/json" https://api.fakecompany.com/v1/tramites/uno2tres  <b>Respuesta:</b>  <pre>HTTP/1.1 400 {   "codigo": 400,   "error": "No se puede obtener el tramite de id uno2tres" }</pre>	
401	Unauthorized	No te conozco, dime quien eres y revisaré tus permisos.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -X DELETE -H "Accept: application/json" -d {"idTramite": "1234"} https://api.fakecompany.com/v1/tramites/  <b>Respuesta:</b>  <pre>HTTP/1.1 401 {   "codigo": 401,   "error": "No se encontró el token de acceso" }</pre>	
403	Forbidden	Tus permisos no son suficientes para acceder a este recurso.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -H "Accept: application/json" https://api.fakecompany.com/v1/tramites/observados  <b>Respuesta:</b>  <pre>HTTP/1.1 403 {   "codigo": 403,   "error": "Tu usuario no puede acceder a este recurso" }</pre>	
404	Not Found	El recurso que estas solicitando no existe.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -H "Accept: application/json" https://api.fakecompany.com/v1/tramites/gratis  <b>Respuesta:</b>  <pre>HTTP/1.1 404 {   "codigo": 404,   "error": "Este recurso no existe" }</pre>	

405	Method Not Allowed	Método no implementado o el método no está soportado o lo relacionado a este recurso no cuenta con permisos de acceso.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -X PUT -H "Accept: application/json" -d {"idTramite":"1235"} https://api.fakecompany.com/v1/tramites/1234  <b>Respuesta:</b>  <pre>HTTP/1.1 405 {   "codigo": 405,   "error": "No existe un procedimiento para actualizar el identificador del tramite" }</pre>	
406	Not Acceptable	No hay nada que enviar que coincida la cabecera Accept-*. Por ejemplo solicita un recurso en XML pero solo está disponible en JSON.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -H "Accept: text/xml" https://api.fakecompany.com/v1/tramites/  <b>Respuesta:</b>  <pre>HTTP/1.1 406 {   "codigo": 406,   "error": "No se puede devolver la información solicitada, parametro incorrecto" }</pre>	

## Server Error

500	Internal Server Error	La solicitud parece correcta, pero un problema ha ocurrido en el servidor. El cliente no puede hacer nada al respecto.
	<b>Ejemplo de Consumo</b>  <b>Solicitud:</b>  CURL -i -X PUT -H "Accept: application/json, Token: rdgggfsdsdd...." -d {"descripción":"Descripcion de ejemplo"} https://api.fakecompany.com/v1/tramites/1234  <b>Respuesta:</b>  <pre>HTTP/1.1 500 {   "codigo": 500,   "error": "Ocurrió un error interno al momento de procesar la información enviada, consulte con el administrador de sistema" }</pre>	

## Mensajes de salidas para errores

Se recomienda utilizar la siguiente estructura JSON para detallar los errores:

```
{
  "codigo": "Código del error",
  "error": "Descripción detallada del error"
}
```

En caso de validación de un formulario con varios campos, la estructura recomendada es<sup>5</sup>:

```
{
  "codigo": "Código del error",
  "errores": [
    {
      "campo": "nombre del campo",
      "valor": "valor del campo",
      "error": "Descripción detallada del error"
    },
    {
      "campo": "nombre del campo",
      "valor": "valor del campo",
      "error": "Descripción detallada del error"
    },
    ...
  ]
}
```

## 7.6. ENCABEZADOS

### Ejemplo de encabezados en los servicios REST

Los URI para los recursos REST de Rule Execution Server soportan algunos campos de cabecera HTTP y parámetros URI genéricos. Las colecciones de recursos soportan el parámetro count, filtros y servicios de paginación.

Todos los recursos REST soportan los siguientes campos de cabecera HTTP y parámetros de URI. Puede combinar cualquiera de los parámetros URI genéricos o de tipo de colección en un recurso. Por ejemplo, puede combinar el parámetro fecha con servicios de transferencia de páginas para visualizar los conjuntos de reglas resultantes por páginas y con la fecha en el formato iso8601.

### Campos de cabecera HTTP

#### Accept

Especifica los tipos de contenido que son válidos en el mensaje de respuesta. Si el servidor no puede responder con el tipo de contenido solicitado, se devolverá el código de estado HTTP 406 No Aceptable. Puede utilizar el parámetro de URI equivalente, en lugar del campo de cabecera HTTP.

---

<sup>5</sup> Estándar para API REST  
<https://www.ctic.gob.bo/etherpad/p/tdf3r4657jhgfK7hjnd7rtNHuN1IQjdg90jT6gTbdf34HF>

## **Accept-Language**

Envía la lista de idiomas que son válidos para el mensaje de respuesta. Por ejemplo, Accept-Language: fr, pt-BR significa que el idioma preferido es francés, pero que el portugués de Brasil también se acepta. Puede utilizar el parámetro de URI equivalente, en lugar del campo de cabecera HTTP.

## **Content-Type**

Indica qué tipo de contenido del cuerpo de entidad se envía al destinatario.

## **X-Method-OverrideX-HTTP-Method-Override**

Indica la operación HTTP que se pasa a través de la solicitud actual. Puede utilizar los parámetros de URI equivalentes, en lugar de los campos de cabecera HTTP<sup>6</sup>.

## **7.7. ESTADO DEL SERVICIO**

Mientras el servicio sea consumido por los clientes este deberá estar disponible, sin embargo por cuestiones técnicas esta puede dejar de estar disponible por mantenimiento u otro aspecto. Para no dejar a los usuarios sin servicio, el proveedor deberá implementar un método de re direccionamiento a otro servidor temporal que atienda las peticiones de los clientes y no solo devolver un código de estado 404 genérico.

En casos extremos el proveedor debe informar a los consumidores de sus servicio el estado del mismo en caso de caídas del servidor, así como también el informar en cuanto este en línea nuevamente.

## **8. ANEXOS**

### **Definiciones:**

#### **Servicio web**

Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma:

- Web Services Protocol Stack: conjunto de servicios y protocolos de los servicios web.
- XML (Extensible Markup Language): formato estándar para los datos que se vayan a intercambiar.
- SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Procedure Call): protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales

#### **Servicio SOAP**

Definición: SOAP (originalmente las siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado

<sup>6</sup> Campos de cabecera HTTP y parámetros URI  
[http://www.ibm.com/support/knowledgecenter/es/SSQP76\\_8.7.1/com.ibm.odm.dserver.rules.res.managing/topics/con\\_res\\_restapi\\_rsrcmng\\_headers.html](http://www.ibm.com/support/knowledgecenter/es/SSQP76_8.7.1/com.ibm.odm.dserver.rules.res.managing/topics/con_res_restapi_rsrcmng_headers.html)

por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

### **Publicación y Descubrimiento (wsi)**

Para la publicación y descubrimiento de servicios web se adopta el WSDL ver 2.0 (Web Services Definition Language) lenguaje basado en XML para describir la funcionalidad que proporciona un servicio Web. Una descripción WSDL (archivo WSDL) de un servicio web proporciona una descripción entendible por la máquina (machine readable) de la interfaz del servicio Web, indicando cómo se debe llamar al servicio, qué parámetros espera, y qué estructuras de datos devuelve<sup>7</sup>.

### **Interfaz de consumo**

Los servicios Web SOAP, o servicios Web "big", utilizan mensajes XML para intercomunicarse que siguen el estándar SOAP (Simple Object Access Protocol), un lenguaje XML que define la arquitectura y formato de los mensajes. Dichos sistemas normalmente contienen una descripción legible por la máquina de la descripción de las operaciones ofrecidas por el servicio, escrita en WSDL (Web Services Description Language), que es un lenguaje basado en XML para definir las interfaces sintácticamente<sup>8</sup>.

### **Servicio REST**

**Definición:** La **Transferencia de Estado Representacional** (en inglés Representational State Transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

Utiliza únicamente XML y HTTP. Cada URL representa un objeto sobre el que puedes realizar POST, GET, PUT y DELETE (las operaciones típicas del HTTP)<sup>9</sup>.

## **8. REFERENCIAS**

Semantic Versioning 2.0.0

<http://semver.org/lang/es/>

REST y el versionado de servicios

<https://www.adictosaltrabajo.com/tutoriales/rest-y-el-versionado-de-servicios/>

Campos de cabecera HTTP y parámetros URI

[http://www.ibm.com/support/knowledgecenter/es/SSQP76\\_8.7.1/com.ibm.odm.dserver.rules.res.managing/topics/con\\_res\\_restapi\\_rsrcmng\\_headers.html](http://www.ibm.com/support/knowledgecenter/es/SSQP76_8.7.1/com.ibm.odm.dserver.rules.res.managing/topics/con_res_restapi_rsrcmng_headers.html)

Ventajas e inconvenientes de API REST para el desarrollo

<https://desarrolloweb.com/articulos/ventajas-inconvenientes-apirest-desarrollo.html>

Simple Object Access Protocol

[https://es.m.wikipedia.org/wiki/Simple\\_Object\\_Access\\_Protocol](https://es.m.wikipedia.org/wiki/Simple_Object_Access_Protocol)

<sup>7</sup> Simple Object Access Protocol  
[https://es.m.wikipedia.org/wiki/Simple\\_Object\\_Access\\_Protocol](https://es.m.wikipedia.org/wiki/Simple_Object_Access_Protocol)

<sup>8</sup> Servicios Web y SOA  
<http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/wholesite.pdf>

<sup>9</sup> Ventajas e inconvenientes de API REST para el desarrollo  
<https://desarrolloweb.com/articulos/ventajas-inconvenientes-apirest-desarrollo.html>



Servicios Web y SOA

<http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/wholesite.pdf>

Transferencia de Estado Representacional

[https://es.wikipedia.org/wiki/Transferencia\\_de\\_Estado\\_Representacional](https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional)

Gestión de descubrimientos utilizando la API de REST

[https://www.ibm.com/support/knowledgecenter/es/SSPLFC\\_7.2.1/com.ibm.taddm.doc\\_721/SDKDevGuide/t\\_cm\\_dbsdk\\_restapi\\_managediscovery.html](https://www.ibm.com/support/knowledgecenter/es/SSPLFC_7.2.1/com.ibm.taddm.doc_721/SDKDevGuide/t_cm_dbsdk_restapi_managediscovery.html)

Estándar para API REST

<https://www.ctic.gob.bo/etherpad/p/tdf3r4657jhgfK7hjnd7rtNHuNI1Qjdg90jT6gTbdf34HF>

Ley de Acceso a la Información Pública

<http://www.gobiernoabierto.gob.sv/pages/ley-de-acceso-a-la-informacion-publica>

Control de versiones del servicio

[https://msdn.microsoft.com/es-es/library/ms731060\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/ms731060(v=vs.110).aspx)

Cómo: Control de versiones del servicio

[https://msdn.microsoft.com/es-es/library/ee816862\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/ee816862(v=vs.110).aspx)

Validación de esquema XML

<http://desktop.arcgis.com/es/arcmap/10.3/tools/conversion-toolbox/xml-schema-validation.htm>

Validación de esquemas XML (XSD) con XmlSchemaCollection

[https://msdn.microsoft.com/es-es/library/516b1xk8\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/516b1xk8(v=vs.110).aspx)

JSON Schema: el XSD de nuestros JSONs

<https://unpocodejava.wordpress.com/2013/01/14/json-schema-el-xsd-de-nuestros-jsons/>

JSON Schema – Software

<http://json-schema.org/implementations.html>

JSON – Esquema

[http://www.es.w3eacademy.com/json/json\\_schema.htm](http://www.es.w3eacademy.com/json/json_schema.htm)

Descubrimiento de servicios Web XML

[https://msdn.microsoft.com/es-es/library/aa735703\(v=vs.71\).aspx](https://msdn.microsoft.com/es-es/library/aa735703(v=vs.71).aspx)

Configuración del descubrimiento de servicios API de Java para servicios web XML

[https://www.ibm.com/support/knowledgecenter/es/SSWLG6\\_6.3.0/com.ibm.sr.doc/twsr\\_polsetjaxws\\_jaxws\\_ic.html](https://www.ibm.com/support/knowledgecenter/es/SSWLG6_6.3.0/com.ibm.sr.doc/twsr_polsetjaxws_jaxws_ic.html)

Modelo de Interoperabilidad del Estado (borrador)

<https://www.ctic.gob.bo/etherpad/p/modelo-de-interoperabilidad-del-estado>