Capítulo ...

APIs de Busca e Publicação Em UDDI

No capítulo anterior sobre o modelo de informação de UDDI é mostrado como são estruturados os dados sobre empresas, negócios e serviços. Agora que entendido como a informação é estruturada, o próximo entendimento é como esses dados são criados e acessados. Este capítulo aborda as APIs que devem ser usadas em UDDI, para operações de publicação e busca de conteúdos de registros. A *UDDI API Specification* descreve as APIs de publicação (*publishing*) e de busca (*inquiry*). Tal especificação descreve os tópicos sobre as **interfaces** para se criar/atualizar e excluir registros, encontrar e buscar registros UDDI, e mais o **protocolo** que é usado para acessar registros UDDI num *site-operador*. As interfaces correspondem as APIs de publicação e busca de serviços. O protocolo corresponde ao uso do SOAP que define as chamadas de funções das APIs por meio de XML.

A API de Publicação

A API de Publicação (*Publishing API*) suporta a operação ***publish*** que habilita empresas a colocarem e atualizarem a informação em um registro UDDI. Por

questões de segurança, esta API tem acesso restrito e o projeto UDDI requer que os nodos operadores implementem um protocolo de autenticação que verifica a identidade do usuário ou de uma organização que criará ou atualizará a informação de um registro. Tal API consiste de funções que provedores de serviço podem usar para criarem, atualizarem ou excluírem informação em registros.

A API de Busca

A API de Busca de serviços (*Inquiry API*) suporta a operação ***find***, que habilita consumidores de serviços a navegarem num sistema UDDI para pesquisar registros de provedores de serviços que oferecem um determinado serviço ou tipo de serviço. Qualquer pessoa pode usar a API de busca para realizar consultas sobre o *UDDI Business Registry* (UBR).

Esta API suporta três moldes de consulta (*query patterns*):

- *browse* – que suporta as cinco estruturas do modelo de informação mostrado no capítulo anterior, permitindo consumidores de serviços realizarem ampla busca de empresas e seus negócios, serviços, *templates* ou **tModel**s. Esta consulta retorna a informação geral (*identification key*, *name* e *description*) pertencente à empresa, ao negócio, ao serviço, ao *template* (informação técnica sobre onde se pode acessar um serviço) ou **tModel** (registro de um tipo de serviço, no sentido de como se pode interagir com o serviço).

- *drill-down* – molde de consulta que obtém uma descrição mais detalhada de um negócio, serviço, *template* ou **tModel**. Este molde é usado em conjunção com o molde *browse*, porque requer uma chave de identificação (*identification key*), obtida durante a operação do molde *browse*, a qual é passada como argumento em um molde *drill-down*. Com este molde, consumidores de serviços podem obter informação mais técnica, tais como, a capacidade de integração de um serviço (*integration capabilities*) e escalabilidade do serviço (*scalability*). Estes aspectos podem ser vistos em [...].

- *Invocation pattern* – este molde consulta a estrutura **bindingTemplate**, que contém informação que os programadores precisam para acessar um serviço na Web. Como a localização de um serviço pode mudar, o molde de consulta *invocation pattern* busca a estrutura **bindingTemplate** para obter a localização corrente do serviço. Consumidores de serviço, tipicamente, usam ferramentas automáticas que consultam a estrutura para o acesso à informação. Como se utilizar essas ferramentas, pode ser visto em [...].

A API UDDI é independente de linguagem de programação. Todas as tecnologias que formam a arquitetura de serviços Web são independentes neste sentido. Tal independência é a maneira com que se tem uma ampla utilização por parte dos desenvolvedores de sistemas com serviços na Web, e é obtida com o uso de XML para descrever a informação e dos XML *Schemas* para a descrição de tipos de dados em registros UDDI.

O Protocolo

Depois de consumidores de serviços descobrirem serviços Web compatíveis com suas necessidades, esses devem se conectar e se comunicar com os sistemas de computação onde estão os serviços desejados. O processo de se conectar e se comunicar com um serviço Web é referenciado como *binding* (ligação) [...].

Na especificação da API UDDI, as interfaces são descritas como estruturas de dados de requisição (*request*) e de resposta (*response*) formatadas em XML. Deste modo o acesso a um registro UDDI corresponde a um envio e o recebimento de dados em XML. Em se tratando de comunicação na Internet, o protocolo apropriado é o SOAP, pois define a chamada de uma função usando XML. Portanto, as chamadas das funções definidas nas APIs UDDI serão chamadas via SOAP. Em outras palavras, a API UDDI é descrita em XML e chamadas para esta API são feitas através do SOAP. Assim, podemos criar um documento WSDL que define as interfaces da API UDDI e a ligação (*binding*) do SOAP para elas. Nesse documento WSDL define-se a localização para um determinado registro UDDI.

SOAP pode ser executado sobre vários protocolos de rede. Entretanto, o acesso à API de publicação é através do protocolo HTTPS, uma variante do protocolo HTTP que usa o *Secure Sockets Level* (SSL) para estabelecer

segurança. Isto é uma exigência da especificação UDDI *Operator* [...]. Para mais informação sobre HTTPS e SSL ver [...] [...] [...].

Por envolver segurança, uma outra exigência definida na especificação é o uso de uma combinação de um **ID** de usuário e uma **senha de identificação** do usuário. Todas as funções de publicação são executadas a partir de credenciais que devem ser passadas com cada requisição. Isto é conseguido pelo fornecimento de uma URL de acesso, para a qual as requisições podem ser enviadas, que usa o protocolo HTTPS, garantindo o envio do ID do usuário e a senha. As funções de busca não necessitam dessas credenciais, sendo assim, abertas para todos os usuários, e obtidas pelo fornecimento de uma URL que use o protocolo HTTP sem precisar de nenhum tipo de *login*.

Publicando em UDDI

Na prática, para se publicar em UDDI, existe a opção de se usar uma **interface baseada na Web** (um browser) ou uma **API programática**. Para o caso de usar uma interface na Web, tem-se meios de se acessar um ambiente de teste, nos quais registros de testes podem ser publicados. IBM e Microsoft, que fazem parte do projeto UDDI inicial, proporcionam URLs seguras, que podem ser acessadas tais como:

- IBM: *https://www-3.ibm.com/services/uddi/testregistry*
- Microsoft: *https://test.uddi.microsoft.com*

Para publicar registros de produção, pode-se usar as URLs:

- IBM: *https://www.ibm.com/services/uddi/protect/publish*
- Microsoft: *https://uddi.microsoft.com/publish*

A maior parte dos usuários com pouco experiência em UDDI devem se iniciar usando uma dessas interfaces baseada na Web.

No entanto, existem certas circunstâncias que exigem acesso direto à API de publicação, mostrada na Tabela ... . Por exemplo, na construção de software de seviços Web que automaticamente registra novos serviços Web, ou quando se necessita uma maneira automática de atualizar a ligação (*binding*) para o acesso via URLs, no caso de ter que haver a recuperação por causa de queda desastrosa no funcionamento do sistema.

Buscando Serviços em UDDI

A Tabela ... mostra os **tipos de elementos** existentes num registro UDDI, e suas funções das APIs definidas para eles. Nesta tabela, a lista de funções não está no seu todo, sendo mostradas, somente, as funções mais importantes. Outras funções podem ser obtidas na especificação *API UDDI Programmer* [...]. As principais funções da API de Publicação UDDI são mostradas na Tabela ... e Tabela ... .

Tabela ... - Tipos de Elementos e suas funções.

Tipo de Elemento	Método Find	Método Get	Método Save	Método Delete
<businessEntity>	`find_business()`	`get_businessDetail()`	`save_business()`	`delete_business()`
<businessService>	`find_service()`	`get_serviceDetail()`	`save_service()`	`delete_service()`
<bindingTemplate>	`find_binding()`	`get_bindingDetail()`	`save_binding()`	`delete_binding()`
<tModel>	`find_tModel()`	`get_tModelDetail()`	`save_tModel()`	`delete_tModel()`

Na Tabela .... , as funções de publicação, permitem **criar/atualizar** e **excluir** registros. A criação e a atualização são realizadas com as funções `save_xxx()`. A exclusão de registros é realizada pelas funções `delete_xxx()`. As funções de busca `find_xxx()` e `get_xxx()` permitem, respectivamente, a descoberta e a obtenção de registros.

Por questões de segurança, as funções de publicação exigem, antes de serem executadas, a obtenção de um *token* de autenticação do usuário. As funções que manipulam esse *token* são apresentadas na Tabela ....

Tabela .... – Obtenção e Liberação de *Token* de Autenticação

Nome da Função	Descrição
`get_authToken ()`	Esta função requer um *token* de autenticação do *site* operador. Esse *token* é requerido para todas as subseqüentes funções save_xxx () e delete_xxx ().
`discard_authToken ()`	Esta função requer que o *token* de autenticação requisitado seja descartado e invalidado.

No modelo orientado-a-serviço, toda comunicação é feita usando XML, materializada em mensagens SOAP. A chamada de uma função, é antes, um

documento XML. Como exemplo, são mostrados exemplos do uso das funções de publicação para **autenticar** usuários, **inserir/atualizar** dados e **apagar/ocultar** dados UDDI.

Autenticando Usuários

A publicação de dados UDDI pode somente ser realizada por usuários autenticados. Cada operador UDDI é livre para implementar seu próprio esquema de autenticação.

Quando enviando as solicitações (requests) `save_xxx ()` e `delete_xxx ()`, a API UDDI requer que essas solicitações incluam o *token* de autenticação. Para obter o *token*, deve-se primeiro fazer uma solicitação `get_authToken`. Essa função requer o ID e a senha do usuário. Como exemplo, mostramos uma solicitação para autenticar um usuário:

```
<get_authToken generic=”1.0” xmlns=”urn:uddi-org:api”
    userID=”bosco@inf.ufsc.br” cred=”OBosco” >
</get_authToken>
```

Caso o usuário não seja reconhecido, o operador retornará um erro E_unknownUser. De outro modo, o site-operador retornará um *token* de autenticação, como por exemplo:

```
<authToken generic=”1.0” xmlns=”urn:uddi-org:api”
    operator=http://uddi.[nome-site-operador].[com/org/edu/gov]>
    <authInfo>1BAAAAAAAHmyB2ylo*pV*pnrFoS4a*IblrgZSlpa
     jYC853wq9HifsbaozLxYpG2Bo;1AAAAAAAAAkZi8QkOJ8BJfnMc
     *HeQCtOTHvu3TdkPEogcauDpvtHyxQGczEE0cj9bdl7C48RryrK
     H7ReaF8OHivQEMltSEhgD8RNhmtOrHFdZoWkANFe*uSLmab4VvA
     FKLHFouvDh3MJ*9VK9YMLl4dg$$
    </authInfo>
</authToken>
```

Todas as chamadas subsequentes para criar/atualizar ou apagar registros requerem o uso deste *token*. Quando do término da publicação, tem-se a opção de descartar o *token*, chamando-se a função `discard_authToken()`. Por exemplo:

```
<discard_authToken generic=”1.0” xmlns=”urn:uddi-org:api” >
    <authInfo>1BAAAAAAAHmyB2ylo*pV*pnrFoS4a*IblrgZSlpa
     jYC853wq9HifsbaozLxYpG2Bo;1AAAAAAAAAkZi8QkOJ8BJfnMc
```

```
    *HeQCtOTHvu3TdkPEogcauDpvtHyxQGczEE0cj9bdl7C48RryrK
    H7ReaF8OHivQEMltSEhgD8RNhmtOrHFdZoWkANFe*uSLmab4VvA
    FKLHFouvDh3MJ*9VK9YMLl4dg$$
   </authInfo>
</discard_authToken>
```

Caso o usuário seja bem sucedido, o site-operador retornará um código de status `E_success`, tal como:

```
<dispositionReport generic="1.0"
    operator=" [nome-site-operador] " xmlns="urn:uddi-org:api">
    <result errno="0">
       <errInfo  errCode="E_success" ></errInfo>
   </result>
</dispositionReport>
```

Se o usuário resolve discartar o *token*, ele sempre poderá obter um novo *token*, chamando a função `get_authToken`, outra vez.

Criando e Salvando Dados UDDI

A API de publicação habilita o usuário a inserir novos registros ou atualizar registros existentes. Para inserir um novo registro, considere usar a função `save_tModel ( )`:

```
<save_tModel  generic="1.0"  xmlns="urn:uddi-org:api" >
    <authInfo>1BAAAAAAAHmyB2ylo*pV*pnrFoS4a*IblrgZSlpa
    jYC853wq9HifsbaozLxYpG2Bo;1AAAAAAAAAkZi8QkOJ8BJfnMc
    *HeQCtOTHvu3TdkPEogcauDpvtHyxQGczEE0cj9bdl7C48RryrK
    H7ReaF8OHivQEMltSEhgD8RNhmtOrHFdZoWkANFe*uSLmab4VvA
    FKLHFouvDh3MJ*9VK9YMLl4dg$$
    </authInfo>
    <tModel tModelKey=" ">
        <name>Interface de Consulta de Preço</name>
        <description  xml:lang="po">
            Interface SOAP para consulta de preço de produtos
        </description>
        <overviewDoc>
            <description  xml:lang="po">
                inserir um arquivo WSDL
            </description>
            <overviewURL>
              http://www.[nome-provedor-serviço].[com/org/edu/gov]/services/
              [nome-arquivo-WSDL].wsdl
```

```
            </overviewURL>
        </overviewDoc>
    </tModel>
</save_tModel>
```

Com este trecho de código XML, pode-se registrar um novo tipo de serviço tModel para a interface de consulta de preços. Porque o atributo tModelKey é vazio, o operador UDDI considera este, uma inserção de registro. O registro tModel é agoara atribuído a seu próprio ID único.

O site-operador retorna, então, a seguinte resposta:

```
<tModelDetail  generic="1.0"  operator="[nome-site-operador]
 truncated="false"  xmlns="urn:uddi-org:api">
    <tModel authorizedName="João Bosco" operator="[nome-site-operador]"
     tModelKey="uuid:01EBBD03-324D-4D7C-97EA—79B9C396D6EA">
        <name>Interface de Consulta de Preço</name>
        <description  xml:lang="po">
            Interface SOAP para consulta de preço de produtos
        </description>
        <overviewDoc>
            <description  xml:lang="po">
                inserir um arquivo WSDL
            </description>
            <overviewURL>
              http://www.[nome-provedor-serviço].[com/org/edu/gov]/services/
              [nome-arquivo-WSDL].wsdl
            </overviewURL>
        </overviewDoc>
    </tModel>
<tModelDetail>
```

Atualizando a Descrição de um Negócio

Suponha agora que, já se tem registrado um negócio, mas deseja-se atualizar a descrição do mesmo. Neste caso, a função `save_business()` deve ser usada para armazenar dados gerais da estrutura *business entity*. Para o protocolo SOAP, o elemento XML apropriado é:

```
<save_business  generic="1.0" xmlns="urn:uddi-org:api">
    <authInfo>1BAAAAAAAHmyB2ylo*pV*pnrFoS4a*IblrgZSlpa
    jYC853wq9HifsbaozLxYpG2Bo;1AAAAAAAAAkZi8QkOJ8BJfnMc
```

```
      *HeQCtOTHvu3TdkPEogcauDpvtHyxQGczEE0cj9bdl7C48RryrK
      H7ReaF8OHivQEMltSEhgD8RNhmtOrHFdZoWkANFe*uSLmab4VvA
      FKLHFouvDh3MJ*9VK9YMLl4dg$$
    </authInfo>
    <businessEntity>
        businessKey=”03754729-3D3C-48E0-854A-1C1FD576CA5B”>
        <name>[nome-provedor-serviço]</name>
        <description  xml:lang=”po”>
            [descrição do negócio] – exemplo: Fornecedor de .....
         </description>
     </businessEntity>
</save_business>
```

Note neste exemplo, o uso do *token* de autenticação e a descrição do negócio atualizada. O valor de businessKey determina que é para atualizar a descrição do negócio. Sendo este atributo, vazio, o que se pretende é uma inserção de um registro de negócio.

Em resposta, o operador UDDI ecoará de volta o novo dado salvo. Por exemplo:

```
<businessDetail  generic=”1.0”  operator=”[nome-site-operador]
 truncated=”false”  xmlns=”urn:uddi-org:api”>
   <businessEntity
      authorizedName=”João Bosco”
      businessKey=”uuid:”03754729-3D3C-48E0-854A-1C1FD576CA5B”
      operator=”[nome-site-operador]”>
      <discoveryURLs>
          <discoveryURL  useType=”businessEntity”>
              http://test.uddi.[nome-siteoperador].[com/org/edu/gov]/
              discovery?businessKey=03754729-3D3C-48E0-854A-1C1FD576CA5
          </discoveryURL>
      </discoveryURLs>
      <name>[nome-provedor-serviço]</name>
      <description  xml:lang=”po”>
          [descrição do negócio] – exemplo: Fornecedor de .....
      </description>
   </businessEntity>
</businessDetail>
```