

Manual do OPENBUS 2.0.0

Tecgraf

6 de março de 2014

Sumário

1	Introdução	1
1.1	Quando Utilizar o OPENBUS	2
2	Arquitetura	2
2.1	Barramento	3
2.2	Serviços Núcleo	5
2.3	Serviços Extras	5
2.4	Biblioteca de Acesso	5
2.4.1	Comunicação Através do Barramento	6
3	Perspectivas	6
3.1	Administradores de Sistema	6
3.2	Gerentes do Barramento	8
3.3	Geração de chaves	10
3.4	Desenvolvedores de Sistemas Integrados	12
4	Glossário	13

1 Introdução

O OPENBUS [8] é um projeto de um middleware para integração de sistemas computacionais heterogêneos, ou seja, desenvolvidos em diferentes linguagens de programação e plataformas computacionais. O OPENBUS se baseia em duas tecnologias complementares para constituir uma infraestrutura básica de integração de sistemas. São elas:

CORBA *Common Object Requester Broker Architecture* [7] é um padrão da indústria que especifica um middleware para sistemas distribuídos heterogêneos orientados a objetos. CORBA define o modelo básico de comunicação usado nas integrações de sistemas feitas com o OPENBUS. CORBA tem suporte para inúmeras linguagens de programação e plataformas computacionais, que nos permite integrar com o OPENBUS uma grande variedade de sistemas.

SCS *Software Component System* [1] é um modelo simples e flexível de componentes de software baseado em CORBA que permite estruturar sistemas usando uma arquitetura baseada em componentes. O SCS é usado no OPENBUS tanto como um modelo arquitetural básico para a infraestrutura básica oferecida como também para estruturar a forma como as integrações são feitas.

Em cima dessas duas tecnologias o OPENBUS introduz duas novas extensões, que basicamente definem a infraestrutura especializada para integração de sistemas computacionais:

Barramento de Integração É o conceito central do OPENBUS. O barramento é o meio através do qual toda interação e comunicação entre os sistemas integrados é feita. O barramento é uma extensão de CORBA com suporte a controle de acesso, que basicamente consiste na autenticação de todo acesso a sistemas através do barramento, permitindo assim identificar de forma segura a origem de toda comunicação (chamadas CORBA) feita através do barramento.

Serviços de Apoio à Integração Juntamente ao barramento, o OPENBUS também provê suporte para registro e descoberta de serviços ofertados pelos sistemas integrados, comunicação baseada em eventos, entre outras funcionalidades através de uma arquitetura orientada a serviços (*Service-Oriented Architecture*, ou SOA [2]). O objetivo desses serviços é oferecer funcionalidades básicas e essenciais que visem facilitar e agilizar o desenvolvimento da integração dos diferentes sistemas.

Este documento tem como objetivo apresentar o OPENBUS 2.0.0 e seus conceitos principais. Nesta seção introdutória apresentaremos uma definição do projeto e os motivadores para o seu uso. Na seção 2 apresentamos a arquitetura do sistema, e entramos um pouco mais em detalhes de como ocorre a comunicação dentro do barramento na seção 2.4.1. Em seguida, na seção 3 falamos um pouco mais sobre o projeto de acordo com a visão de cada tipo de usuário, e, por fim, na seção 4 apresentamos um glossário com os conceitos principais do projeto.

1.1 Quando Utilizar o OpenBus

O OPENBUS é um middleware para integração de sistemas heterogêneos. Contudo, a integração de sistemas pode se dar de diversas formas diferentes e envolver requisitos diversos. Por exemplo, uma forma de integração extremamente simples entre dois sistemas que apenas precisam trocar dados é por meio da troca de arquivos num formato adotado por ambos sistemas. Inclusive esses arquivos podem ser transmitidos pela rede manualmente ou automaticamente.

Em outros cenários, a integração pode também exigir alguma colaboração entre os sistemas, através da execução comandos, por exemplo. Nesse caso, é necessário que os sistemas forneçam alguma forma para receber comandos, que pode ser através de mensagens enviadas por um socket, comandos de um Webservice, chamadas a um objeto remoto, etc. Fazer um sistema implementar inúmeras interfaces de acesso para integração com diferentes sistemas é uma solução pouco razoável num cenário em que devam existir vários sistemas a serem integrados. Idealmente, deve-se adotar uma tecnologia de comunicação genérica e eficiente o suficiente para ser adequada ao uso com sistemas com diferentes requisitos.

Outra necessidade comum na integração de sistemas cooperativos é o controle de acesso e a governança das integrações. Ou seja, quando os sistemas integrados não são abertos ao acesso público e irrestrito, é necessário que a infraestrutura forneça mecanismos que permitam restringir quais serviços podem ser integrados e como essas integrações podem ser feitas.

O OPENBUS oferece uma infraestrutura adequada para implementar integrações entre sistemas tendo essas questões e necessidades em mente. Em particular, o OPENBUS se baseia na tecnologia CORBA para definir uma tecnologia de comunicação genérica e eficiente para integração de sistemas escritos em diferentes linguagens de programação e plataformas computacionais. Além disso, o OPENBUS estende a tecnologia CORBA com suporte a um rigoroso controle de acesso que permite a inspeção e controle das integrações através de um modelo de governança, onde um gerente do barramento pode controlar quais sistemas acessam o barramento e se integram a outros sistemas.

2 Arquitetura

A infraestrutura mínima do OPENBUS é representada pelo seu núcleo principal, que é composto por um barramento de comunicação e serviços essenciais, denominados serviços núcleo. Além do núcleo do barramento, a arquitetura OPENBUS também define um conjunto de elementos adicionais que fornecem outras facilidades importantes, tais como bibliotecas e serviços extras.

O diagrama ilustra a arquitetura do OpenBus, dividida em duas seções principais: **Serviços Núcleo** (topo) e **Serviços Extras** (fundo), conectadas por um **Barramento = CORBA + Controle de Acesso** (centro).

Serviços Núcleo:

- Um **Registro de Ofertas** (caixa cinza) está conectado ao barramento.
- Três **Serviço** (caixas azuis) estão conectados ao barramento através de suas respectivas **Biblioteca de Acesso** (caixas brancas).

Serviços Extras:

- Um **Serviço de Colaboração** (caixa verde) está conectado ao barramento.
- Dois **Aplicação** (caixas azuis) estão conectados ao barramento através de suas respectivas **Biblioteca de Acesso** (caixas brancas).

O **Núcleo do OpenBus** é a interface central que gerencia o acesso ao barramento.

2.1 Barramento

Essas entidades são tipicamente sistemas computacionais e usuários desses sistemas. Cada entidade é identificada por um nome único no barramento a ser definido pelo gerente barramento. Tipicamente, os nomes de entidade são nomes de sistemas computacionais que oferecem serviços no barramento e nomes de contas de usuários numa base de diretórios como o LDAP.

3

acessa o barramento simultaneamente, é interessante que cada processo possa utilizar um login próprio para acesso ao barramento.

O OPENBUS define três formas para autenticação de uma entidade no processo de criação de logins de acesso. Essas três formas de autenticação são:

Autenticação por Senha Neste caso, a autenticação é feita através de uma senha fornecida juntamente com o nome da entidade. A senha é validada por um módulo de validação de senhas especificado pelo gerente do barramento. Tipicamente esse módulo de validação é integrado a alguma base de dados de usuário que fornece informações para validação das senhas. Essa forma de autenticação é geralmente utilizada para incorporar ao barramento um grande número de usuários mantidos num sistema separado. Um exemplo dessa integração é o validador de senhas LDAP fornecido pelo OPENBUS, que permite autenticar nomes de usuários num serviço de diretórios LDAP como entidades para acesso ao barramento. Neste caso, todos os nomes de usuários na base LDAP são automaticamente entidades autorizadas a acessar o barramento.

Autenticação por Certificado Neste caso, a autenticação é feita através de um certificado previamente cadastrado pelo gerente do barramento em nome de uma entidade particular. Para que a autenticação ocorra, é preciso decodificar um desafio encriptado com a chave pública contida no certificado cadastrado. Para tanto, é necessário ter a chave privada correspondente ao certificado cadastrado. Essa forma de autenticação é geralmente utilizada para autorizar o acesso de sistemas computacionais específicos que fornecem serviços através do barramento. Toda gerência dos certificados de acesso cadastrados no barramento é de responsabilidade do gerente do barramento, que utiliza ferramentas fornecidas pelo OPENBUS para adicionar, remover e consultar os certificados cadastrados.

Autenticação Compartilhada Neste caso, a autenticação é feita em colaboração com outro sistema que já possua um login no barramento, ou seja, já está autenticado em nome de uma entidade para acessar o barramento. Neste caso, o sistema já autenticado produz um segredo a ser compartilhado com o outro sistema que é utilizado na autenticação desse novo sistema em nome da entidade autenticadora do sistema original. Essa forma de autenticação é geralmente utilizada quando um sistema deseja compartilhar sua forma de autenticação com outro sistema sem fornecer informações privilegiadas como senhas ou chaves privadas.

Tipicamente os logins de acesso ao barramento ficam válidos por um período denominado *lease*. Após o período de *lease*, o login deve ser renovado para que possa continuar válido¹. Independentemente disso, todo login pode se tornar inválido a revelia da aplicação. Neste caso, uma nova autenticação deve ser realizada para obtenção de um novo login para continuar acessando o barramento². Uma vez inválido, um login nunca volta a ser válido novamente.

O barramento possui algumas características importantes que devem ser mencionadas. Uma delas, é que o barramento persiste todo o seu estado no diretório de dados. Dessa forma, sempre que ele é iniciado, e o mesmo já possui uma base de dados populada, esses dados serão recarregados e farão parte do estado inicial do barramento.

Outra característica importante é que o barramento mantém compatibilidade com a a versão imediatamente anterior do barramento. Ou seja, mesmo que a versão do barramento evolua, as entidades ainda conseguem acessar o barramento utilizando bibliotecas de acesso de uma versão anterior. O OPENBUS utiliza um esquema de versionamento com quatro números, no formato *A.B.C.D*, onde:

- Campos **A** e **B** significam a versão da IDL e deve ser igual em todos os pacotes que são compatíveis: versão IDL, barramento e bibliotecas de acesso.
- Campo **C** é incrementado quando ocorre alguma modificação que não altera a IDL do núcleo do barramento.

¹Essa tarefa de renovação de login é feita automaticamente pela biblioteca de acesso do OPENBUS.

²A identificação de que o login ficou inválido e o restabelecimento do login podem ser feitos através da callback `onInvalidLogin` da biblioteca de acesso do OPENBUS.

- Campo **D** representa uma versão apenas com correções de falhas.

Sendo assim, o barramento 2.0 permite a realização de acesso utilizando as bibliotecas de acesso de versão 1.5.x, onde x pode assumir qualquer valor de *C.D.* Porém, é importante deixar claro que a versão 2.0 traz muitas melhorias no quesito de segurança, e essas melhorias não serão usufruídas por clientes que acessam o barramento com versões antigas das bibliotecas de acesso. O barramento permite que clientes 1.5 se comuniquem com clientes 2.0, e vice-e-versa, porém essas comunicações se realizam com o mesmo nível de segurança que existia na versão 1.5 do barramento.

2.2 Serviços Núcleo

O barramento oferece em seu núcleo um serviço essencial denominado Registro de Ofertas. Através dele é possível publicar, buscar e monitorar os demais serviços disponíveis através do barramento.

Todo serviço a ser ofertado através do Registro de Ofertas deve ser implementado como um componente SCS [1] que oferece diferentes facetas através das quais o serviço é acessado. No momento do registro de uma oferta de serviço, deve ser fornecido um conjunto pares nome e valor que devem descrever propriedades particulares do serviço sendo ofertado. Essas propriedades poderão ser utilizadas como critério de filtro no momento de buscar por ofertas de serviço no Registro de Ofertas.

Além das propriedades fornecidas no registro da oferta, o próprio serviço de Registro de Oferta também gera um conjunto de propriedades automáticas que descrevem:

- login com que a oferta foi registrada.
- nome da entidade que registrou a oferta.
- momento do registro da oferta.
- nome e versão do componente SCS que implementa o serviço.
- facetas e interfaces do component SCS que implementa o serviço.

O Registro de Ofertas também disponibiliza um mecanismo de observação de publicação, atualização e remoção de ofertas. Maiores detalhes sobre a especificação e o uso do serviço de Registro de Ofertas podem ser encontrados nos manuais de uso das bibliotecas de acesso (seção 2.4)

2.3 Serviços Extras

A versão atual do OPENBUS oferece um serviço extra denominado Serviço de Colaboração. Esse serviço tem como principais funcionalidades:

- Criar e compartilhar uma sessão de colaboração.
- Oferecer um canal de comunicação para enviar eventos para todos os membros da sessão.

Não entraremos em maiores detalhes sobre as funcionalidades e o uso do Serviço de Colaboração neste documento. Para mais informações, consulte a documentação do próprio serviço [14].

2.4 Biblioteca de Acesso

O OPENBUS também disponibiliza uma biblioteca de acesso ao barramento a ser utilizada na implementação da integração de sistemas ao barramento. Essa biblioteca basicamente implementa uma API simplificada para acesso ao barramento, ocultando detalhes do protocolo de comunicação do OPENBUS. Através dessa biblioteca é possível realizar as seguintes operações, dentre outras:

- Autenticação de entidades;

- Obtenção de logins;
- Renovação automática de *lease* de logins;
- Receber notificação de o login se tornou inválido;
- Realizar chamadas utilizando diferentes logins;
- Identificar os logins e entidades que originaram cada chamada recebida.

A biblioteca de acesso do OPENBUS tem implementações nas quatro diferentes linguagens de programação suportadas oficialmente pelo OPENBUS, em particular, Java [12], C# [10], C++ [11] e Lua [13].

2.4.1 Comunicação Através do Barramento

O barramento é internamente implementado como uma arquitetura orientada a serviços (SOA). Em princípio, todas as chamadas feitas através do barramento envolvem chamadas a esses serviços internos, em particular, o serviço de Controle de Acesso.

Para evitar a sobrecarga dessas chamadas adicionais a serviços internos do barramento, a biblioteca de acesso implementa otimizações eficientes que permitem reduzir drasticamente essa sobrecarga nos cenários de uso mais comuns. Em particular, a biblioteca de acesso faz uso de vários caches internos para minimizar a necessidade de chamadas adicionais a serviços internos da implementação do barramento.

Em cenários de integração típicos, a maior parte da comunicação através do barramento é feita diretamente entre os sistemas integrados, sem necessidade de acesso à infraestrutura interna do barramento. Essa característica da biblioteca de acesso traz duas vantagens importantes:

- Primeiramente, o desempenho da comunicação entre dois sistemas só depende da qualidade da rede de comunicação entre esses dois sistemas.
- Além disso, mesmo que a infraestrutura do barramento fique indisponível por alguma falha inesperada, a comunicação entre os serviços pode continuar, mesmo que com um nível de qualidade limitado.

3 Perspectivas

Agora que já apresentamos a arquitetura e os conceitos gerais do OPENBUS, iremos falar um pouco sobre algumas características específicas do OPENBUS, que dependem do papel que o usuário tem no barramento. Os papéis abordados são:

Administrador de Sistema As pessoas que possuem acesso à máquina onde o barramento será executado e são os responsáveis por levantar e parar o barramento.

Gerente do Barramento Aqueles que irão desempenhar o papel de organizar e acompanhar o que é publicado no barramento.

Desenvolvedor de Sistemas Integrados Responsáveis por implementar a integração de seus respectivos serviços e aplicações ao barramento.

3.1 Administradores de Sistema

O barramento e os serviços núcleo são distribuídos em um mesmo programa, o *busservice*. Logo, para disparar o barramento, basta executar o programa passando as suas configurações por linha de comando ou por arquivo de configuração. Se uma configuração não for explicitada, será utilizado o seu valor padrão.

As opções de configuração são:

- host** Endereço de rede usado pelo barramento.
Valor padrão é “*”, que significa que vai ouvir em todos os IPs da máquina.
- port** Número da porta usada pelo barramento.
Valor padrão é 2089.
- database** Arquivo de dados do barramento.
Valor padrão é “openbus.db”.
- privatekey** Arquivo com chave privada do barramento.
Valor padrão é “openbus.key”
- leasetime** Tempo em segundos de lease dos logins de acesso.
Valor padrão é 1800 segundos.
- expirationgap** Tempo em segundos que os logins ficam válidos após o lease.
Valor padrão é 10 segundos.
- admin** Usuário com privilégio de administração.
Não é definido um valor padrão.
- validator** Nome de pacote de validação de login.
Não é definido um valor padrão.
- loglevel** Nível de log gerado pelo barramento.
O valor padrão é 3. Os níveis vão de 0 a 5, onde 5 é o nível com mais detalhes, e o 0 desativa o log.
- logfile** Arquivo de log gerado pelo barramento.
Por padrão o log é direcionado para a saída padrão (*standard output*).
- oilloglevel** Nível de log gerado pelo OiL [6, 5] (debug).
O valor padrão é 0. Os níveis vão de 0 a 5, onde 5 é o nível com mais detalhes, e o 0 desativa o log.
- oillogfile** Arquivo de log gerado pelo OiL (debug).
Por padrão o log é direcionado para a saída padrão (*standard output*).
- noauthorizations** Desativa o suporte à autorizações de oferta.
- nolegacy** Desativa o suporte à versão antiga do barramento.
- configs** Arquivo de configurações adicionais do barramento.
O valor padrão é “openbus.cfg”.

As opções “admin” e “validator” podem receber mais de um valor. Quando essas opções são configuradas por linha de comando, basta repetir a opção o número de vezes desejadas. Quando são configuradas através do arquivo de configuração, é necessário informar a lista de valores desejados, no formato de uma tabela LUA [4]. O código 1 apresenta um exemplo de arquivo de configuração.

Código 1: Exemplo de arquivo de configuração.

```
1 validator = {"openbus.test.core.services.TesterUserValidator"}
2 loglevel = 5
3 admin = {"admin1", "admin2"}
4 leasetime = 30
```

Existem duas formas de se definir o arquivo de configuração que o programa irá utilizar: utilizando a opção “config” por linha de comando, ou definindo um valor para a variável de ambiente *OPENBUS_CONFIG*.

Porém, as opções passadas por linha de comando têm prioridade sobre as opções definidas no arquivo de configuração.

Através da opção “`validator`” informa-se o módulo do validador de senhas que será utilizado pelo barramento. Disponibilizamos em `openbus.core.services.passwordvalidator.LDAP` um validador LDAP, que precisa ser configurado com a lista de servidores utilizados na autenticação. Para maiores informações sobre como configurar o validador LDAP, consulte [9].

Também é possível desabilitar o suporte de compatibilidade com a versão anterior através da opção “`no-legacy`”. Ao ativar essa opção, o barramento deixa de permitir que entidades se autenticuem no barramento utilizando as bibliotecas de acesso de versão 1.5.x.

3.2 Gerentes do Barramento

Para desempenhar o papel de gerente do barramento, disponibilizamos a ferramenta *busadmin*, que permite que se realize atividades de governança (administração) sobre o barramento.

Antes de entrar em detalhes sobre o uso e as funcionalidades do *busadmin*, vamos agora apresentar alguns conceitos importantes de governança dentro do OPENBUS. Existem os conceitos de:

Categoria Representa uma categoria de entidades no barramento. Categorias de entidade são agrupamentos usados exclusivamente para facilitar a gerência das diversas entidades cadastradas no barramento pelo administrador.

Entidade Representa uma entidade do barramento registrada. Entidade é tudo aquilo que pode fazer login no barramento e usufruir dos seus recursos. Em particular, tanto usuários como implantações de sistema são considerados entidades. Entidades podem ou não ser cadastradas no barramento, mas apenas entidades cadastradas podem ser autorizadas a ofertar serviços.

Certificado Chave pública que pode ser usada para autenticar uma dada entidade no barramento. Ver seção sobre geração de chaves para informações sobre como gerar as chaves pública e privada. É possível adicionar certificados para entidades não cadastradas no barramento.

Interface Definição de uma interface IDL de um serviço que pode ser ofertado no barramento.

Autorização Associação de uma interface IDL a uma entidade, indicando que processos conectados como essa entidade podem oferecer serviços que implementem essa interface no Registro de Ofertas do barramento.

Para utilizar a ferramenta, deve-se executar:

```
busadmin [opções] --login=<usuário> <comando>
```

Para adicionar e remover categorias, entidades, certificados, interfaces e autorizações, aconselhamos que se defina um arquivo de script LUA com o formato especificado no código 2. Ele serve de entrada para os comandos “`script`” e “`undo-script`” da ferramenta *busadmin*, que, respectivamente, executa e desfaz a execução do lote de comandos especificados pelo arquivo de script.

O *busadmin* também permite realizar esses cadastros e descadastros manualmente, além de realizar consultas e outras atividades de gerência do barramento. A listagem completa dos comandos disponíveis é apresentada a seguir:

- **Opções:**

- host=<endereço> Informa o endereço do Barramento.

- Valor padrão é 127.0.0.1.

- port=<porta> Informa a porta do Barramento.

- Valor padrão é 2089.

- verbose=<nível>** Aciona o verbose da API OPENBUS.
Valor padrão é 0. Os níveis vão de 0 a 5, onde 5 é o nível com mais detalhes, e o 0 desativa o log.
- oilverbose=<nível>** Aciona o verbose do OIL.
Valor padrão é 0. Os níveis vão de 0 a 5, onde 5 é o nível com mais detalhes, e o 0 desativa o log.
- certificate=<arquivo>** Informa a chave privada para realizar a autenticação por certificado, ao invés de ser por senha. O padrão é realizar a autenticação por senha, onde a ferramenta pergunta a senha antes de executar o comando.

- **Controle de Categoria:**

- add-category=<id> --name=<nome>** Adiciona uma categoria com o identificador e nome descritivo especificado.
- del-category=<id>** Remove a categoria com o identificador especificado.
- set-category=<id> --name=<nome>** Altera o nome descritivo da categoria com o identificador especificado.
- list-category** Mostra as informações de todas as categorias cadastradas.
- list-category=<id>** Mostra informações da categoria especificada.

- **Controle de Entidade:**

- add-entity=<id> --category=<id_categoria> --name=<nome>** Adiciona uma entidade com o identificador, categoria e nome descritivo especificado.
- del-entity=<id>** Remove a entidade com o identificador especificado.
- set-entity=<id> --name=<nome>** Altera o nome descritivo da entidade com o identificador especificado.
- list-entity** Mostra as informações de todas as entidades cadastradas.
- list-entity=<id>** Mostra informações da entidade especificada.
- list-entity --category=<id_categoria>** Mostra informações das entidades pertencentes a categoria especificada.

- **Controle de Certificado:**

- add-certificate=<id_entidade> --certificate=<certificado>** Cadastra o certificado para a entidade especificada.
- del-certificate=<id_entidade>** Remove o certificado da entidade especificada.

- **Controle de Interface:**

- add-interface=<interface>** Adiciona a interface na lista de interfaces permitidas no barramento.
- del-interface=<interface>** Remove a interface da lista de interfaces permitidas no barramento.
- list-interface=<interface>** Mostra as interfaces permitidas no barramento.

- **Controle de Autorização:**

- set-authorization=<id_entidade> --grant=<interface>** Autoriza a entidade a publicar a interface especificada.
- set-authorization=<id_entidade> --revoke=<interface>** Remove a autorização da entidade de publicar a interface especificada.
- list-authorization** Mostra todas as autorizações concedidas no barramento.

--list-authorization=<id_entidade> Mostra todas as autorizações da entidade especificada.
--list-authorization --interface="<interface1> <interface1>...<interfaceN>" Mostra todas as autorizações contendo as interfaces especificadas.

- **Controle de Ofertas:**

--del-offer Lista todas as ofertas publicadas no barramento e aguarda a escolha de uma oferta para ser removida.
--del-offer --entity=<id> Lista todas as ofertas publicadas pela entidade especificada e aguarda a escolha de uma oferta para ser removida.
--list-offer Lista todas as ofertas publicadas no barramento.
--list-offer=<id> Lista todas as ofertas publicadas pela entidade especificada.
--list-props Lista todas as ofertas publicadas no barramento e aguarda a escolha de uma oferta para listar todas as suas propriedades.
--list-props=<id> Lista todas as ofertas publicadas pela entidade especificada e aguarda a escolha de uma oferta para listar todas as suas propriedades.

- **Controle de Logins:**

--del-login=<id> Remove o login especificado.
--list-login Mostra todos os logins ativos no barramento.
--list-login --entity=<id> Mostra todos os logins ativos da entidade especificada.

- **Script:**

--script Executa um script LUA com um lote de comandos.
--undo-script Desfaz a execução de um script LUA com um lote de comandos.

- **Relatório:**

--report Constrói um relatório sobre o estado atual do barramento.

3.3 Geração de chaves

Dentro do pacote de distribuição do OpenBus, está incluso o binário do OpenSSL, que permite gerar chaves privada e pública. Antes de utilizar o binário, é necessário executar os seguintes comandos (estamos utilizando comandos do Bash shell [3] no exemplo abaixo):

```
#este comando é apenas para facilitar a legibilidade do código abaixo, sendo opcional
export OPENBUS_HOME=<local de extracao do pacote>

export LD_LIBRARY_PATH="${OPENBUS_HOME}/lib:${LD_LIBRARY_PATH}"

# caso seja MacOS é preciso também:
export DYLD_LIBRARY_PATH="${OPENBUS_HOME}/lib:${DYLD_LIBRARY_PATH}"
```

Para gerar a chave privada, utilize os seguintes comandos:

```

${OPENBUS_HOME}/bin/openssl genrsa -out tmp_openssl.key 2048
${OPENBUS_HOME}/bin/openssl pkcs8 -topk8 -nocrypt \
    -in tmp_openssl.key -out <nome do par de chaves>.key -outform DER
rm -f tmp_openssl.key

```

Para gerar o certificado, utilize o seguinte código:

```

${OPENBUS_HOME}/bin/openssl req -config $OPENSSL_HOME/openssl.cnf -new -x509 \
    -key <nome do par de chaves>.key -keyform DER \
    -out <nome do par de chaves>.crt -outform DER

```

Código 2: Exemplo de script para a ferramenta *busadmin*.

```

1 — Definição de uma categoria
2 — * comando: Category
3 — * parâmetros:
4 —   * id = identificador da categoria
5 —   * name = descrição da categoria
6 Category {
7   id = "TEST.Category",
8   name = "Descrição da categoria",
9 }
10 — Definição de uma entidade
11 — * comando: Entity
12 — * parâmetros:
13 —   * id = identificador da entidade
14 —   * category = identificador da categoria que a entidade pertence
15 —   * name = descrição da entidade
16 Entity {
17   id = "TEST.Entity",
18   category = "TEST.Category",
19   name = "Descrição da entidade",
20 }
21 — Definição de um certificado
22 — * comando: Certificate
23 — * parâmetros:
24 —   * id = identificador da entidade
25 —   * certificate = caminho para arquivo de certificado associado a entidade
26 Certificate {
27   id = "TEST.Entity",
28   certificate = "teste.crt",
29 }
30 — Definição de uma interface
31 — * comando: Interface
32 — * parâmetros:
33 —   * id = replID da interface
34 Interface {
35   id = "IDL:script/Test:1.0"
36 }
37 — Conceder autorização
38 — * comando: Grant
39 — * parâmetros:
40 —   * id = identificador da entidade a ser autorizada
41 —   * interfaces = lista de interfaces a serem autorizadas.
42 Grant {
43   id = "TEST.Entity",
44   interfaces = {
45     "IDL:script/Test:1.0",
46   }
47 }
48 — Remover autorização
49 — * comando: Revoke
50 — * parâmetros:
51 —   * id = identificador da entidade
52 —   * interfaces = lista de interfaces a serem desautorizadas.
53 Revoke {
54   id = "TEST.Entity",
55   interfaces = {
56     "IDL:script/Test:1.0",
57   }
58 }

```

Para converter uma chave utilizada em um SDK 1.5 para uma chave que funcione com SDKs 2.0, utilize o seguinte comando:

```
${OPENBUS_HOME}/bin/openssl pkcs8 -topk8 -nocrypt -in chave.pem -inform PEM -out chave.der -outform DER
```

3.4 Desenvolvedores de Sistemas Integrados

Como informado na seção 2.1, oferecemos uma biblioteca de acesso, que implementa o protocolo e exporta uma API de programação, a qual oferece alguns facilitadores e auxilia a interação com o barramento. Para maiores informações, consulte o manual de uso da biblioteca de acesso (seção 2.4).

4 Glossário

A

API *Application Program Interface*. Interface de programação oferecida às aplicações que precisam acessar o barramento, seja para fazer ou receber chamadas através do barramento, ou acessar serviços oferecidos pelo barramento.

Autorização Associação de uma interface IDL a uma entidade, indicando que processos autenticados como essa entidade possam oferecer serviços que implementem essa interface no Registro de Ofertas do barramento.

B

Barramento Toda infraestrutura oferecida pelo OPENBUS que permite fazer chamadas CORBA com a identificação das entidades com que os processos que originaram a chamada foram autenticados.

Biblioteca de Acesso Biblioteca de programação que implementa a API do OPENBUS. O projeto OPENBUS oferece implementações dessa biblioteca nas linguagens Lua, Java, C++ e C#.

busadmin Ferramenta que permite realizar operações de administração do barramento (governança).

busservices Programa que implementa o barramento e os serviços núcleo do barramento OPENBUS.

C

Categoria Representa uma categoria de entidades no barramento. Categorias de entidade são agrupamentos usados exclusivamente para facilitar a gerência das diversas entidades cadastradas no barramento pelo administrador.

Certificado Chave pública que pode ser usado para autenticar uma dada entidade no barramento.

Controle de Acesso Serve como ponto de entrada do barramento, sendo responsável por autenticar, renovar e gerenciar os logins de serviços e aplicações ao barramento.

CORBA *Common Object Request Broker Architecture*. Especificação de um padrão de ORB sobre a qual o OpenBus é definido e implementado. O OPENBUS pode ser visto como uma extensão do padrão CORBA.

E

Entidade Qualquer coisa que pode ser autorizada a acessar o barramento. Cada entidade possui um identificador único na forma de um nome. Entidades podem ser usuários humanos de sistemas ou mesmo instalações de serviços específicos de aplicações.

I

IDL *Interface Description Language*. Linguagem de descrição de tipos e interfaces de CORBA.

Interface Definição de uma interface IDL de um serviço que pode ser ofertado no barramento.

L

Lease Período de tempo pelo qual um login no barramento permanecerá válido sem necessidade de renovação. Contudo, um login pode ser invalidado explicitamente por um administrador antes do tempo de lease.

LDAP *Lightweight Directory Access Protocol*. É um protocolo de Internet para acessar serviços de diretório distribuídos. É comumente usado para autenticação em redes corporativas.

Login Representa uma autenticação de uma entidade junto ao barramento para poder acessá-lo.

Logout Processo no qual o criador de um login o invalida, fazendo com que esse login não possa mais ser usado para acessar o barramento.

O

Oferta de Serviço Cadastro no Registro de Ofertas de um componente SCS que implementa um conjunto de interfaces autorizadas que representam um dado serviço a ser utilizado através do barramento.

OpenBus Nome deste projeto que define um barramento baseado em CORBA para integração de aplicações corporativas multilinguagem.

P

Propriedades da Oferta Conjunto de pares nome e valor que descrevem uma oferta de serviço.

Protocolo Conjunto de regras de comunicação para acesso ao barramento OPENBUS.

R

Registro de Ofertas Serviço núcleo do barramento que permite registrar e buscar ofertas de serviços no barramento. Esse serviço é disponibilizado através da API.

S

SCS *Software Component System*. Modelo de componentes de software distribuído adotado pelo OPENBUS. Todos serviços ofertados no barramento devem ser modelados como componentes SCS.

Serviços Núcleo Serviços oferecidos pelo próprio barramento, tipicamente acessados através da API.

Serviços Extras Serviços que acrescentam funcionalidades para auxiliar a integração entre serviços e aplicações, mas que não são parte do núcleo do barramento.

V

Validadores LDAP São utilizados para autenticar usuários em serviços de diretório que utilizam o protocolo LDAP.

Referências

- [1] C. Augusto, E. Fonseca, L. Marques, S. Correa, and R. Cerqueira. SCS: Software component system. <http://www.tecgraf.puc-rio.br/scs>, May 2006.
- [2] T. Erl. *Service-oriented architecture: concepts, technology, and design*. Prentice Hall PTR, 2005.
- [3] GNU. Bash reference manual. <http://www.gnu.org/software/bash/manual/bashref.html>, 2010.
- [4] Roberto Ierusalimsky. The programming language lua. <http://www.lua.org/>, February 2008.
- [5] Renato Maia. OiL: An object request broker in the Lua language. <http://www.tecgraf.puc-rio.br/~maia/oil/>, 2005.
- [6] Renato Maia, Renato Cerqueira, and Ricardo Calheiros. OiL: An object request broker in the Lua language. In Mauro S. P. Fonseca, editor, *Proceedings of SBRC'06 - Salão de Ferramentas*, pages 1439–1446, Porto Alegre, Brazil, June 2006. SBC.
- [7] Object Management Group. *The Common Object Request Broker Architecture (CORBA) Specification - Version 3.1*, January 2008. document: formal/2008-01-04.
- [8] TecGraf. OpenBus - Enterprise Integration Application Middleware. <http://www.tecgraf.puc-rio.br/openbus>, 2006.
- [9] TecGraf. *Manual de instalação do OpenBus 2.0.0*. TecGraf, 2012.
- [10] TecGraf. *Tutorial do SDK C# 2.0.0*. TecGraf, 2012.
- [11] TecGraf. *Tutorial do SDK C++ 2.0.0*. TecGraf, 2012.
- [12] TecGraf. *Tutorial do SDK Java 2.0.0*. TecGraf, 2012.
- [13] TecGraf. *Tutorial do SDK Lua 2.0.0*. TecGraf, 2012.
- [14] TecGraf. Documentação do Serviço de Colaboração 1.0. <https://jira.tecgraf.puc-rio.br/confluence/pages/viewpage.action?pageId=52528899>, 2013.