Apple Push Notification Service (APNS)[1]

O Apple Push Notification Service (APNS) é a peça central para o envio de notificações do tipo *push* para dispositivos iOS. Essas notificações são enviadas a partir de servidores de aplicação de terceiros, passando por um servidor intermediário da Apple, até chegar ao dispositivo móvel de destino.

Uma notificação *push* é uma mensagem curta, composta por duas partes principais: o *token* do dispositivo destinatário e o *payload.* O *token* do dispositivo é um identificador do equipamento que permite ao APNS autenticar o envio da notificação. Já o *payload,* consiste de uma lista de propriedades (no formato JSON) que especifica como o usuário da aplicação será alertado.

O provedor das notificações deve conectar-se ao APNS a partir de um canal persistente e seguro. É através deste canal que as notificações devem ser enviadas aos servidores do *Apple Push Notification Service*, estes realizarão o encaminhamento das notificações aos dispositivos destinatários.

Fluxo de envio

O fluxo de envio de notificações remotas é uma via única (servidor-dispositivo). O provedor empacota a notificação, que inclui o *token* do dispositivo e o *payload*, e a envia para o APNS; este, por sua vez, encaminha a notificação aos dispositivos destinatários. Ver Figura 1.

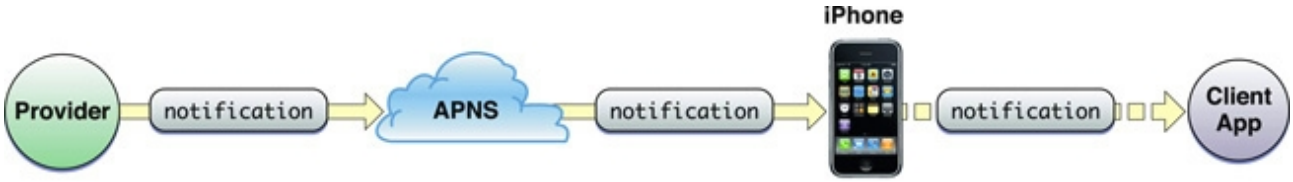

Figura 1: Fluxo de envio das notificações push.

Notificação recusada

Se a notificação for recusada (para os casos do aplicativo ter sido desinstalado ou o usuário tenha revogado a permissão para receber as notificações) o APNS informa ao servidor provedor da notificação através do serviço de *feedback.* O serviço de *feedback* disponibiliza também quais dispositivos têm recusado as notificações, desta forma o provedor poderá excluí-los da lista de envio.

Qualidade de serviço

O serviço de notificação *push* da Apple, por padrão, oferece o a função de "*store-and-forward*". Essa função permite que, se o dispositivo estiver offline, a notificação seja armazenada no APNS e, assim que o dispositivo se conectar, ser enviada.

Vale destacar que essa funcionalidade de "*store-and-forward*" não garante a entrega da informação, uma vez que o APNS não tem controle sobre a conexão do dispositivo. As mensagens armazenadas no APNS são armazenadas por um tempo limitado e, após esse tempo, é descartada. Não foi encontrado na documentação da Apple o tempo de armazenamento dessas mensagens.

Importante frisar ainda que apenas uma notificação (por aplicativo em um dispositivo) é

1 Extraído e adaptado de: http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html

armazenada. Desta forma, sempre a última notificação será enviada ao dispositivo.

Arquitetura de Segurança

O serviço de notificação *push* proposto pela Apple, se baseia em dois níveis de confiança: "*connection trust*" e "*token trust*".

Connection trust: garante que o APNS está enviando a notificação para um dispositivo válido e que o dispositivo está recebendo a notificação de um APNS legítimo.

- **Service-to-Device Connection Trust:** (1) o dispositivo inicia uma conexão peer-to-peer através de uma TLS, (2) o servidor APNs envia seu certificado ao dispositivo, (3) o dispositivo valida o certificado do servidor, (4) o dispositivo envia seu certificado ao servidor APNs, (5) o APNs valida o certificado do dispositivo, (6) a conexão é estabelecida. Ver Figura 2.

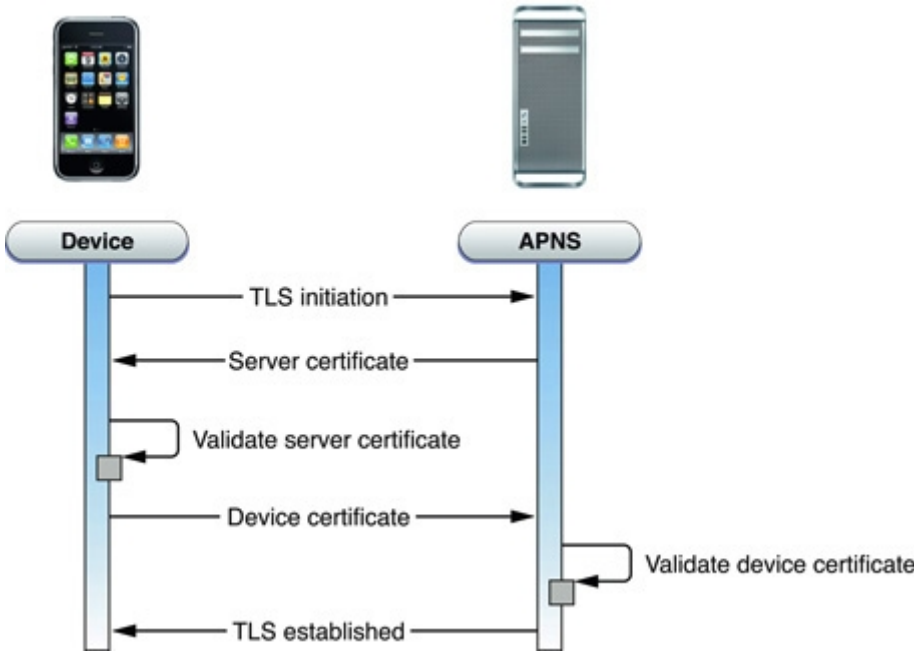

Figura 2: Service-to-Device Connection Trust

- **Provider-to-Service Connection Trust:** procedimento semelhante ao *Service-to-Device Connection Trust* . Ver Figura 3.

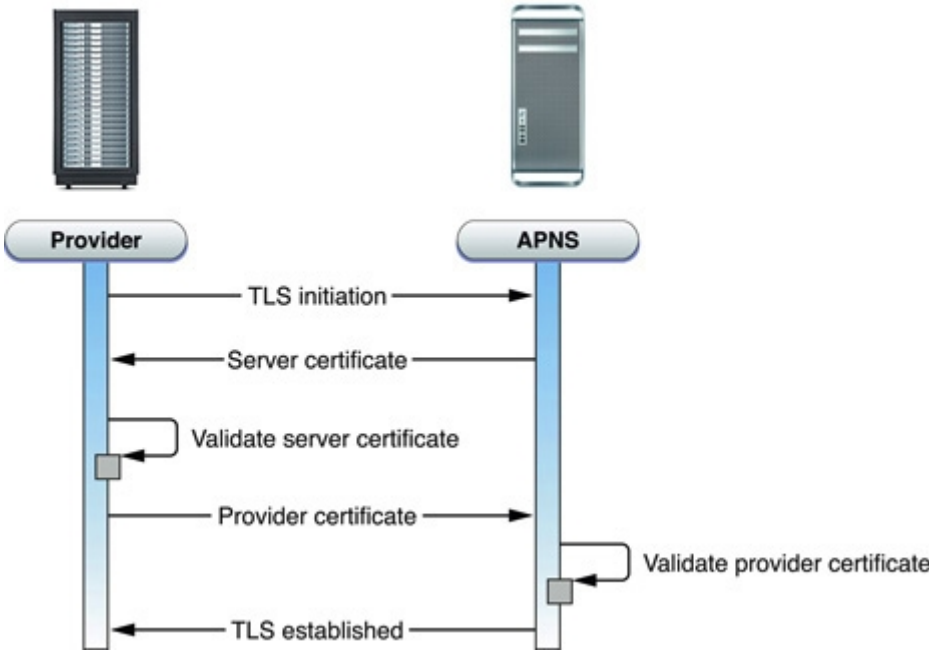

Figura 3: Provider-to-Service Connection Trust.

Obs.: o tratamento para essa comunicação e estabelecimento de conexão é realizado pelo sistema operacional, não sendo necessário qualquer implementação no aplicativo.

Token Trust: Toda notificação enviada pelo provedor para o APNS deve estar acompanhada do *token* do dispositivo. Ao receber a notificação, o APNS faz a validação usando a chave para descriptografar. Se for uma notificação válida, encaminha ao dispositivo. Ver Figura 4.

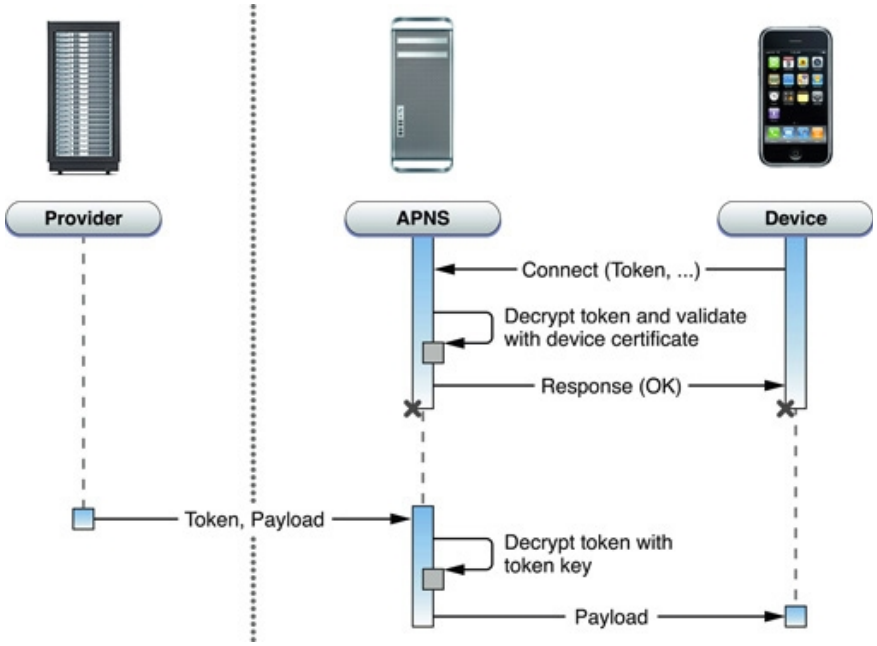

Figura 4: Token Trust.

O payload da notificação

O tamanho máximo das informações (*payload*) de uma notificação é 256 bytes. Notificações que excedam esse limite são recusadas. Para cada notificação, o provedor deve compor um objeto "JSON dictionary" (RFC 4627).

Os campos existentes em uma notificação são:

- uma mensagem de alerta para ser apresentada ao usuário;
- o número que será apresentado no "balão" do ícone do aplicativo ; e
- um som para ser tocado .

Componentes envolvidos no APNS

- Um **dispositivo com iOS ou Mac OS** e a aplicação instalada.
- Um **servidor de aplicação** de terceiro que envia os dados ao dispositivo através do servidor APNS da Apple.
- **Servidor APNS da Apple** envolvido na recepção de mensagens do servidor de aplicação e no seu envio ao dispositivo.

Credenciais envolvidas no APNS

- **Certificado do remetente:** certificado digital que identifique o servidor provedor da notificação no APNS.
- **Certificado do servidor APNS:** certificado digital que identifique e garanta a legitimidade do servidor de APNS.
- **ID do dispositivo**: o identificador do dispositivo iOS ou Mac OS que está registrado para receber mensagens via APNS.
- **Token do dispositivo:** identificador do dispositivo e da aplicação que receberá as notificações do APNS.

Passos para criar um Servidor de Push Notification

Este conteúdo foi reproduzido de: https://github.com/exmo/pushServerIOS

1 - Criar uma aplicação java EE

Opatamos por criar uma aplicação java Demoiselle, mas você pode criar da forma como preferir.

Como criamos com o demoiselle utilizei no pom do projeto o parent demoiselle-servlet-parent, pois desta forma o projeto já é compatível com JEE 6 e não traz dependências extras...

2 - Cria uma classe REST que vai enviar o Push para a Apple.

Criamos a classe PushNotificationRS, que terá o método push:

```
	@GET
	@Path("/{token}/{msg}")
	@Produces("application/json; charset=UTF-8")
	public String[] push(@PathParam("token") String token, @PathParam("msg") String msg)
		ApnsService service = getService();
			String payload = APNS.newPayload().alertBody(msg).badge(10).sound("message").build();
			service.push(token, payload);
			return new String[]{"Token: "+token, "MSG: "+msg};
	  }
```

Este método apenas recebe o token do dispositivo a a mensagem a ser enviada.

A comunicação com o servidor da apple(APNS) se dá através de sockets sobre SSL e as informações trafegam no formato JSON. Como não queremos reinventar a roda utilizamos a biblioteca Java-APNS, que vai simplificar muito nossos trabalhos.

Omiti o método getService por enquanto, pois ele envolve a proxima etapa que é a obtenção do certificado e chave privada.

3 - Obtenção do Certificado e chave privada do servidor.

A conexão é feita sobre SSL, logo teremos que ter um certificado combinado com a apple.

Sigam os passos fornecidos pela Apple para gerar o certificado

Após seguir estes passos, você terá um arquivo com extensão .p12 que conterá o certifica e a chave privada. Adicione este arquivo no classpath do seu projeto. Em nosso caso colocamos um arquivo chamado "ServerCertificadoEchavePrivada.p12"

Agora podemos ver o método getService que apenas abre este arquivo com a senha utilizada em sua criação:

```
	public ApnsService getService(){
		InputStream path =
PushNotificationRS.class.getClassLoader().getResourceAsStream(CERTIFICADO);
		ApnsService service =
				APNS.newService()
				.withCert(path, PASSWORD)
				.withSandboxDestination()
				.build();
		return service;
	}
```

4 - Envie as mensagens e teste em seu dispositivo!

Rode o servidor, rodamos no jboss 7. Em seguida abra o browser e execute uma url como esta:

"http://localhost:8080/push/rest/push/4f2330c0f36336f02aad1dca8e2d453b15bde23dcf49307a89ebfa5d3ee161c6/Minha mensagem push"

Referências

- Apple Developer
- raywenderlich : Apple Push Notification Services Tutorial
- Apple Developer: Certificados! A parte mais difícil

Passos para criar o projeto cliente

Este conteúdo foi reproduzido de: https://github.com/exmo/pushClientIOS

1 - Criamos um projeto do tipo Single View (pode ser qualquer um...)

2 - Modificamos a classe AppDelegate para registrar o aplicativo na apple e obter o token que representará o dispositivo.

Registrando o dispositivo

```
    - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions{
          [[UIApplication sharedApplication] registerForRemoteNotificationTypes:
(UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeSound | UIRemoteNotificationTypeAlert)];
          return YES;
    }
```

Recebendo o token (neste momento deve-se guardar o token ou envia-lo para a aplicação associa-lo a um cpf por exemplo)

```
    - (void)application:(UIApplication*)application didRegisterForRemoteNotificationsWithDeviceToken:
(NSData*)deviceToken{
            NSLog(@"My token is: %@", deviceToken);
    }
```

*Obs: Nesta classe tem um método utilitário para formatar o token...

3 - Assinar o aplicativo com o perfil criado para esta aplicação (isso esta melhor explicado na doc do server)

O perfil utilizado para o push deve ser o mesmo que foi configurado para o aplicativo e dispositivos. Também deve ser o mesmo utilizado para o servidor.

Somente é possível testar o push em um dispositivo, no simulador não funciona.

Referências:

- Apple Developer
- raywenderlich : Apple Push Notification Services Tutorial