5
Comandos de Edição NCL

Como discutido na Seção 3.1.2, o núcleo do ambiente de apresentação Ginga-NCL é composto pelo Formatador NCL e o módulo Gerente de Bases Privadas. O Formatador NCL é responsável por receber uma aplicação NCL e controlar sua apresentação, tentando garantir que as relações entre os objetos de mídia, especificadas pelo autor da aplicação, sejam respeitadas. O Formatador lida com documentos NCL que são coletados dentro de uma estrutura de dados conhecida como base privada. O Gerente de Base Privada é responsável por receber e tratar comandos para gerenciamento das bases privadas, comandos para controle do ciclo de vida de aplicações e comandos de edição ao vivo das aplicações. Este capítulo discute o suporte a esses comandos, genericamente chamados de comandos de edição NCL. Este capítulo descreve apenas a sintaxe e semântica dos diversos comandos que servirá de base para as discussões a respeito da identificação de recursos e controle do ciclo de vida das aplicações, apresentadas nos Capítulos 6 e 7, respectivamente.

5.1.
Bases Privadas e Comandos de Edição

Um comando de edição pode ser recebido de diversas formas: pelo canal de TVD sintonizado, por outras redes, por exemplo, pelo canal de interatividade, ou por eventos gerados por objetos imperativos (por exemplo, eventos especificados em Lua) parte das aplicações NCL. As formas com que os comandos de edição podem ser envelopados e transportados são discutidas no Capítulo 8.

Os comandos de edição NCL são divididos em três grupos:

- Gerência de bases privadas: comandos para operações em bases privadas (para abrir, ativar, desativar, fechar e salvar bases privadas). Esse grupo de comandos de edição é apresentado na Tabela 4;
- Controle do ciclo de vida de documentos NCL: comandos para manipulação de documentos NCL em bases privadas (para adicionar,

remover e salvar um documento em uma base privada aberta e para iniciar, pausar, retomar e parar apresentações de documentos em uma base privada ativa). Esses comandos são apresentados na Tabela 5;

- Edição de entidades NCL: comandos para manipular entidades NCL em uma base privada aberta. Para cada entidade NCL, foram definidos os comandos *add* e *remove*. Se uma entidade já existir, o comando *add* tem a semântica de atualização (alteração). Desse grupo de comandos de edição, apenas o comando *addNode* é apresentado na Tabela 6, por ser importante nos algoritmos de identificação de recursos discutidos no Capítulo 6 desta tese. Uma tabela com todos os comandos de edição de entidades NCL é apresentada no Anexo II.

Na Tabela 4, Tabela 5 e Tabela 6 os comandos de edição são definidos na coluna "Comando", onde, cercados por parênteses e separados por vírgulas, os parâmetros necessários a cada comando são apresentados. Na coluna "*tag*" são apresentados os valores em hexadecimal que identificam univocamente o comando de edição. As funcionalidades dos comandos são apresentadas na coluna "Descrição".

Comando	*tag*	Descrição
openBase (baseId, location)	0x00	Abre uma base privada existente localizada pelo parâmetro *location*. Se a base privada não existir ou se o parâmetro *location* não for informado, uma nova base é criada com o identificador *baseId*. O parâmetro *location* especifica o dispositivo de armazenamento no ambiente de recepção e o caminho da base a ser aberta.
activateBase (baseId)	0x01	Ativa uma base privada aberta. As aplicações de uma base privada ativa podem ser iniciadas.
deactivateBase (baseId)	0x02	Desativa uma base privada aberta. Todas as aplicações da base têm sua apresentação finalizada.
saveBase (baseId, location)	0x03	Salva todo o conteúdo da base privada em um dispositivo de armazenamento persistente (se disponível). O parâmetro *location* especifica o dispositivo e caminho para salvar a base.
closeBase (baseId)	0x04	Fecha a base privada aberta e descarta todo o conteúdo da base privada.

Tabela 4 – Comandos de edição para operações sobre bases privadas.

Como mencionado no Capítulo 3, o Gerente de Bases Privadas associa pelo menos uma base privada a um canal de TVD (conjunto de serviços[15]). Quando um

[15] Definição de serviço pode ser encontrada no Anexo 1.

canal é sintonizado, sua base privada *default* é aberta e ativada pelo Gerente de Bases Privadas; todas as outras bases privadas são desativadas. Uma base privada pode ser criada ou aberta através de comandos recebidos pelo canal sintonizado. No entanto, pode existir apenas uma base privada associada a um serviço do canal sintonizado. Se um canal de TVD possuir apenas um serviço, existirá apenas uma base privada associada ao canal, que consiste na base privada *default*. Nesse caso, qualquer comando de edição para abrir ou criar uma base privada, recebido pelo canal de TVD sintonizado, é ignorado.

A base privada associada ao canal de TVD possui como identificador (na Tabela 4, parâmetro *baseId*) o valor do identificador do fluxo de transporte recebido. Caso sejam criadas as bases privadas associadas a um serviço do canal sintonizado, elas possuem o seguinte *baseId*: o valor do identificador de rede do canal (Morris, 2005), seguido da string ".", concatenado com o valor do identificador do serviço. Por exemplo, no caso de uso das tabelas SI, como no SBTVD, as tabelas NIT e PMT possuem, respectivamente, o identificador de rede do canal (*network_id*) e o identificador do serviço (*program_number*).

As aplicações NCL residentes[16] são gerenciadas em uma base privada específica para aplicações residentes.

Por razões de segurança, apenas uma única base privada pode estar ativa por vez. O modo mais simples e restritivo de gerenciar bases privadas é ter uma única base privada aberta por vez. Assim, se o usuário mudar o canal selecionado, a base privada atual seria fechada. Nesse caso, o comando *openBase* é sempre seguido pelo comando *activeBase* e o comando *deactiveBase* nunca é usado. Contudo, o número de bases privadas que podem ser mantidas abertas é uma decisão de implementação do middleware.

É importante ressaltar que as bases criadas por comandos de edição recebidos pelo canal de TVD sintonizado, incluindo a base privada *default*, podem ser controladas apenas por comandos de edição recebidos por esse mesmo canal. Além disso, os objetos imperativos, dispostos em uma base privada controlada por comandos de edição recebidos pelo canal de TVD sintonizado, podem gerar eventos com comandos de edição para controlar apenas essa base.

[16] Uma aplicação residente consiste em uma aplicação armazenada no ambiente de recepção e que pode ter sua apresentação iniciada em qualquer instante.

Os documentos NCL em uma base privada podem ser iniciados, pausados, retomados e parados e podem referir-se uns aos outros. No Capítulo 7 é discutido como esses comandos de edição são utilizados para realizar o controle do ciclo de vida das aplicações NCL.

Comando	*tag*	Descrição
addDocument (baseId, {uri, id}+)	0x05	Adiciona um documento NCL a uma base privada aberta. Os arquivos do documento NCL podem ser: a) enviados sem solicitação pela rede de difusão de dados; nesse caso, o par {uri, id} é usado para relacionar um conjunto de caminhos de arquivos especificados no documento NCL com suas respectivas localizações no sistema de transporte (veja Capítulo 6); b) recebidos pelo canal de interatividade sob demanda, ou já ser residentes no receptor; para esses arquivos, nenhum par {uri, id} necessita ser enviado, exceto o par {uri, "null"} associado ao documento NCL, que deverá ser adicionado na base baseId, se o documento NCL não for recebido sem solicitação (pushed file).
removeDocument (baseId, documentId)	0x06	Remove um documento NCL, identificado pelo parâmetro documentId, de uma base privada aberta
saveDocument (baseId, documentId, location)	0x2E	Salva um documento NCL de uma base privada aberta em um dispositivo de armazenamento persistente (se disponível). O parâmetro location deve especificar o dispositivo e o caminho no dispositivo onde o documento será salvo. Se o documento NCL estiver sendo exibido, ele deve primeiro ser parado (todos os eventos no estado occurring devem ser parados)
startDocument (baseId, documentId, interfaceId, offset, nptBaseId, nptTrigger) O parâmetro offset especifica um valor de tempo.	0x07	Inicia a reprodução de um documento NCL em uma base privada ativa, iniciando a apresentação a partir de uma interface específica do documento (interfaceId). A referência do tempo transportada no campo nptTrigger estabelece o ponto de início do documento documentId, com respeito à base temporal NPT identificada por nptBaseId. Três casos podem ocorrer: 1) Se nptTrigger for diferente de 0 e for maior ou igual ao valor atual de NPT da base temporal identificada por nptBaseId, espera-se até que NPT atinja o valor dado em nptTrigger para iniciar a apresentação do documento do seu ponto inícial no tempo+offset; 2) Se nptTrigger for diferente de 0 e for menor que o valor atual de NPT da base temporal identificada por nptBaseId, o início da apresentação do documento é imediato e deslocado no tempo de seu ponto inicial do valor "offset+(NPT – nptTrigger)segundos"; NOTA: Somente nesse caso, o parâmetro offset pode receber um valor negativo, mas offset+(NPT – nptTrigger)segundos é sempre um valor positivo. 3) Se nptTrigger for igual a 0, a exibição do documento é imediata e a partir de seu ponto inicial no tempo + offset.
stopDocument (baseId, documentId)	0x08	Pára a apresentação de um documento NCL em uma base privada ativa. Todos os eventos do documento que estão em andamento são parados.
pauseDocument (baseId, documentId)	0x09	Pausa a apresentação de um documento NCL em uma base privada ativa. Todos os eventos do documento que estão em andamento são pausados.

resumeDocument (baseId, documentId)	0x0A	Retoma a apresentação de um documento NCL em uma base privada ativa. Todos os eventos do documento que foram previamente pausados pelo o comando de edição pauseDocument são retomados.

Tabela 5 – Comandos de edição para manipulação de documentos NCL.

Note pela Tabela 5 que o comando *addDocument* faz uso de pares {uri,id} para identificar recursos. A mesma identificação pode ser observada na Tabela 6 para o comando *addNode*. No Capítulo 6 é discutido como esses parâmetros são utilizados para que os recursos referenciados pelas aplicações sejam identificados de forma apropriada.

Comando	***tag***	**Descrição**
addNode (baseId, documentId, compositeId, {uri, id}+)	0x27	Adiciona um nó (elemento <media>, <context> ou <switch>) a um nó de composição (elemento <body>, <context> ou <switch> identificado por compositeId) de um documento NCL em uma base privada aberta. A especificação XML do nó e seu conteúdo de mídia podem: a) ser enviados sem solicitação pela rede de difusão de dados; nesse caso, o par {uri, id} é usado para relacionar um conjunto de caminhos de arquivos, definidos no documento XML da especificação do nó, com suas respectivas localizações no sistema de transporte; b) recebidos pelo canal de interatividade sob demanda, ou já ser residentes no receptor; para esses arquivos, nenhum par {uri, id} necessita ser enviado, exceto o par {uri, "null"} associado à especificação XML do nó NCL que deverá ser adicionado em compositeId, caso o documento XML não seja recebido sem solicitação (pushed file)

Tabela 6 – Comandos de edição addNode.

As entidades NCL utilizadas nos comandos de edição devem ser um documento em conformidade com o perfil de Comando NCL 3.0, definido por um esquema XML[17]. Conforme esse esquema XML, diferentemente dos documentos NCL, diversos elementos NCL podem ter o elemento-raiz nos parâmetros de comando XML.

5.2. Implementação

A parte central do suporte aos comandos de edição implementado no Ginga-NCL é o Gerente de Bases Privadas, apresentado na Figura 8 como *PrivateBaseManager*. O componente *PrivateBaseManager* possui um ouvinte do componente *Tuner*, que implementa a interface *ITunerListener*. Quando o *PrivateBaseManager* é instanciado, esse ouvinte é cadastrado no componente

[17] http://www.ncl.org.br/NCL3.0/profiles/NCL30EdCommand.xsd

Tuner através da interface *ITuner*. Sempre que um canal de TVD é sintonizado, o *PrivateBaseManager* cria a base privada *default* desse canal. Para isso, o *PrivateBaseManager* solicita o identificador do fluxo de transporte do canal sintonizado ao componente *DataProcessor*, através da interface *IDataProcessor*.

Ainda durante a instanciação do *PrivateBaseManager*, uma base privada para aplicações residentes, com identificador "residentApps", é aberta. Na versão atual da implementação, essa base privada possui a aplicação residente descrita em (Moreno, 2009c). Além disso, na instanciação do *PrivateBaseManager*, três tipos de ouvintes de comandos de edição são criados:

- Ouvinte de comandos de edição recebidos pelo canal de TVD sintonizado: implementa a interface *IStreamEventListener* e é cadastrado no componente *DataProcessor*, através da interface *IDataProcessor*;
- Ouvinte de comandos de edição recebidos pelos demais canais de interatividade: implementa a interface *IEventListener* e é cadastrado no componente *TransportManager*, através da interface *ITransportManager*;
- Ouvinte de comandos de edição recebidos por eventos gerados por objetos imperativos: implementa a interface *IApplicationPlayerListener* e é cadastrado no componente *Player*, através da interface *IApplicationPlayer*.

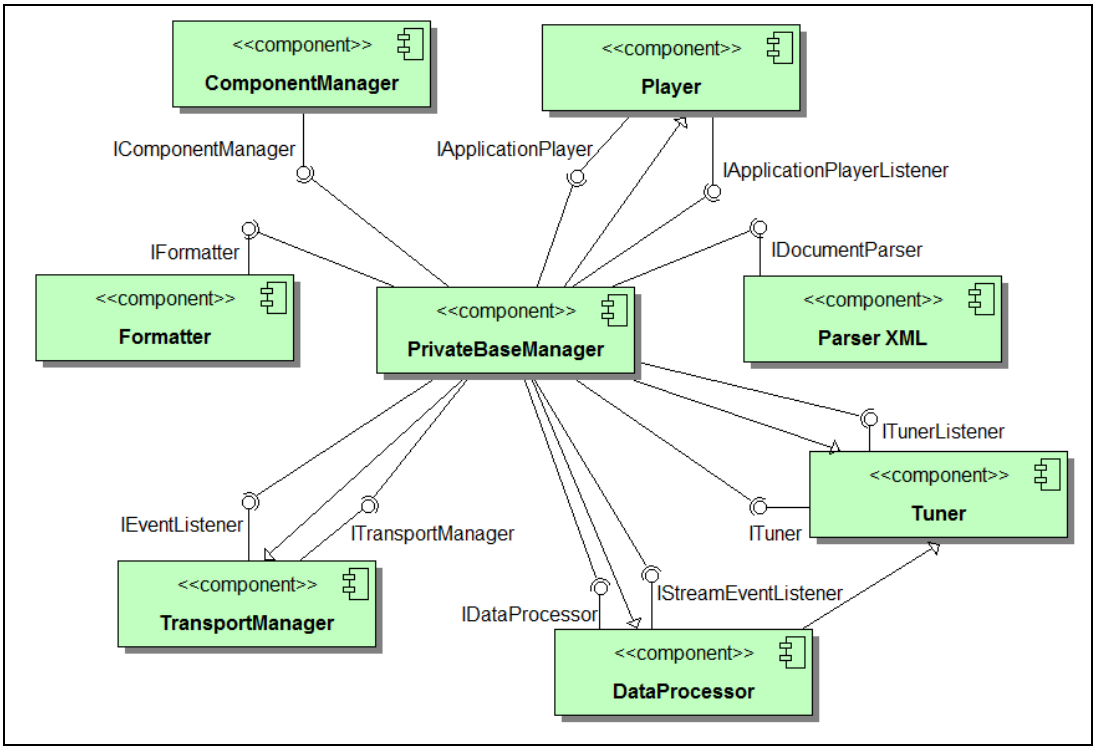

Figura 8 – Diagrama de Componentes para Tratamento dos Comandos de Edição.

Por meio desses três ouvintes, o componente *PrivateBaseManager* é notificado sempre que um comando de edição encontra-se pronto para ser processado (como discutido no Capítulo 8). Como a notificação é originada de tipos diferentes, o *PrivateBaseManager* é capaz de diferenciar a origem de um comando de edição. Para os comandos recebidos do canal de TVD sintonizado, o *PrivateBaseManager* pode solicitar ao *DataProcessor*, através da interface *IDataProcessor*, os identificadores de cada serviço e o número de serviços disponíveis no canal.

Ao processar um comando de edição, mudanças sobre as estruturas presentes em uma base privada podem ser realizadas pelo *PrivateBaseManager*. O *PrivateBaseManager* faz com que as mudanças realizadas nas estruturas de uma base privada sejam refletidas na apresentação da aplicação, através da interface *IFormatter* do componente *Formatter*. Para comandos que possuem parâmetros XML, o *PrivateBaseManager* solicita ao *Parser* XML, através da interface *IDocumentParser*, que as especificações XML sejam interpretadas e traduzidas em estruturas de dados que representam as entidades do modelo NCM (Soares, 2005).