RestTL

O RestMS é uma API de troca de mensagens baseada no padrão de desenho REST (1). Esta API assenta sobre a especificação RestTL – Rest Transport Layer- que utiliza HTTP/HTTPS para fornecer o meio de comunicação utilizado pela API RestMS. A RestTL define a utilização de dois headers HTTP, sendo eles:

* When-Modified-After - GET evented on resource date change
* When-None-Match - GET evented on resource ETag change

A especificação RestTL define a utilização dos seguintes métodos HTTP:

* POST - create a new resource
* GET - retrieve a resource
* PUT - update a resource
* DELETE - delete a resource

Os headers HTTP permitidos são os seguintes:

* Location - URI of newly created resource
* ETag - opaque hash tag of that identifies a resource instance
* Data-Modified - date and time that resource was modified
* If-Modified-Since - conditional GET using resource date
* If-None-Match - conditional GET using resource ETag
* If-Unmodified-Since - conditional PUT or DELETE using resource date
* If-Match - conditional PUT or DELETE using resource ETag

Os recursos públicos da API têm URIs baseados no conteúdo, enquanto que os privados têm URIs gerados dinamicamente.

Existem recursos estruturados em que a sua estrutura é conhecida, podendo, inclusivamente, ter os URIs de recursos a que estão ligados. Existem também os recursos opacos, dos quais a estrutura é desconhecida sendo tratados como conteúdo binário, categorizados por um tipo MIME.

Os recursos estruturados são representados como simples documentos XML ou JSON.

Esquema de URI de recursos publicos:

http://{server name}[:{port}]/{schema}/{resource type}/{resource name}

* server name - name of host as for HTTP request
* port - optional port as for HTTP request
* schema - name of the API
* resource type - name of the resource type
* resource name - name of the resource itself

Esquema de nomes de recursos privados:

http://{server-name}[:{port}]/{schema}/resource/{resource hash}

Utilização de cache na obtenção de recursos.

Consistência de actualização de recursos, utilizando os headers HTTP If-Match e If-Unmodified-Since. Quando se tenta actualizar um recurso, quando este foi modificado desde a última verificação, é originado um erro e o recurso não é actualizado.

Consistência de remoção de recursos, utilizando os headers HTTP If-Match e If-Unmodified-Since. A remoção de recursos, provoca a remoção dos recursos que estão abaixo destes na árvore de recursos.

Formatação de tipos MIME:

* " application/{schema}+xml"
* " application/{schema}+json"
* " text/xml”

Mecanismos para comunicação assíncrona:

Asynclet – Utilizando um URI que ainda não exista, o cliente obtém um recurso asynclet, quando o recurso é disponibilizado, o próprio recurso é obtido.

Streaming de recursos para o cliente, utilizando multipart content.

Notificação de alterações de recursos, através da utilização de headers http

A utilização de Asynclet é comum nos casos de filas de mensagens onde os clientes podem apagar ou obter mensagens. Nestes casos é possível que uma fila fique sem mensagens e neste caso será entregue um recurso com o URI do próximo recurso a ser criado, mas sinalizado do facto de ser um Asynclet, ou seja, desse URI ainda não ter nenhum recurso associado. Apenas quando o recurso for de facto criado, será retornado para o cliente. Aquando a recepção de um recurso podem ser recebidos vários recursos, incluindo um asynclet. ??? Bloqueio no caso do envio de um Asynclet?

A notificação de alterações é feita utilizando os headers HTTP When-Modified-After e When-None-Match. Quando o recurso não foi actualizado, a resposta do servidor irá reflectir esse facto.

Caching

Header HTTP Cache-Control :

Não autorizado

Cliente

Servidores proxy

Manipulação de cache utilizando headers HTTP  If-Modified-Since e If-None-Match.

ETags representam univocamente cada recurso, atingindo o nivel de detalhe do tipo de representação do recurso.

Idempotencia :

Todos os métodos excepto POSTs de recursos privados, uma vez que os URIs destes são gerados de forma dinâmica.

É possível a utilização de pipelining desde que utilizando operações idempotentes.

RestMS

O RestMS é um serviço de mensagens RESTful que providencia comunicação assíncrona através duma interface REST, utilizando HTTP/HTTPS. O RestMS assenta sobre RestTL e é extensível através de especificações de perfis.

# Domains

Os domains são namespaces para pipes e feeds. Um servidor pode ter vários domains e cada um deles pode ter credenciais de segurança diferentes. Um domain contém os profiles que nele podem ser utilizados

Esquema de URI de domains:

http://{server-name}[:{port}]/restms/domain/{domain-name}

Operações válidas em domains:

* GET - retrieves the domain representation.
* POST - creates a new feed or pipe within the domain.

O domain lista todos os recursos públicos que lhe pertencem, feeds e pipes.

# Profiles

Contém um conjunto de semânticas que o servidor implementa, estas semânticas podem definir o tipo de feeds, pipes, códigos de resposta, utilizações dos objectos definidos.

# Feeds

As feeds são o ponto de entrada de mensagens no sistema. Vários clientes podem escrever mensagens numa feed, a ordem das feeds deve ser mantida, relativamente a cada cliente. As feeds podem ser criadas dinamicamente por clientes.

Operações possíveis sobre feeds:

GET - retrieves the feed.

PUT - updates the feed. The feed name and type cannot be modified.

DELETE - deletes the feed.

POST - sends a message to the feed or stage a content on the feed.

Forma de uma feed:

<?xml version="1.0"?>

<restms xmlns="http://www.restms.org/schema/restms">

<feed

name="{feed name}" mandatory feed name

[ type="{feed type}" ] optional type

[ title="{short title}" ] optional title

[ license="{license name}" ] optional license name

/>

</restms>

O tipo da feed define o tipo de mensagens que esta podem receber e a semântica com que estas são encaminhadas.

# Pipes

Os pipes consistem na forma de recepção de mensagens dos clientes. Os pipes são read-only e são ordenados para um leitor. São criados de forma dinâmica e são privados para os clientes que os criam. O servidor pode limpar pipes que não sejam utilizados ou que estejam a ficar sobrecarregados de mensagens.

Operações possíveis sobre pipes:

GET - retrieves the pipe representation.

DELETE - deletes the pipe.

POST - creates a new join for the pipe.

Descrição de um pipe:

<?xml version="1.0"?>

<restms xmlns="http://www.restms.org/schema/restms">

<pipe

name="{pipe name}" server-generated hash

type="{pipe type}" actual pipe type

[ title="{short title}" ] title, if specified

>

[ <join href="{join URI}"

type="{join type}"

address="{address pattern}"

feed="{feed URI}" /> ] ...

[ <message

href="{message URI}"

address="{address literal}"

message\_id = "{identifier}" /> ] ...

[ <message href="{message URI}" async="1" /> ]

</pipe>

</restms>

O tipo do pipe define a semântica de encaminhamento utilizada. Os pipes não devem ser partilhados e o resultado de uma partilha é imprevisível. Os pipes devem ser reutilizados entre sessões para evitar a perda de mensagens não processadas numa sessão anterior.

# Joins

Os joins definem o encaminhamento que deve ser feito das mensagens recebidas nas feeds para os pipes subscritos por clientes. São criados de forma dinâmica e devem sempre ser privados. A criação de joins deve ser feita sempre para o URI do pipe respectivo. Se a feed ou o pipe a que o join diz respeito forem destruídos, o join também o será.

Operações possíveis sobre joins:

GET - retrieves the join representation.

DELETE - deletes the join.

A especificação de um join

<?xml version="1.0"?>

<restms xmlns="http://www.restms.org/schema/restms">

<join

[ type="{join type}" ] optional type

address="{address pattern}"

feed="{feed URI}" the feed to pull from

>

[ <header name="{header name}" value="{header value}" /> ] ...

</join>

</restms>

O tipo do join define a semântica de selecção de mensagens a encaminhar.

# Mensagens e conteúdos

As mensagens e conteúdos estão ligados, sendo que uma mensagem é um envelope que pode conter conteúdos, nela embebidos, ou endereços para conteúdos sobre a forma de recursos. Estes recursos são especificados com um MIME Type e são tratados como blobs.

Publicação:

Os recursos são primeiro publicados para uma feed, sendo enviados através de pedidos POST, em que o corpo do POST contém a mensagem e o Content-Type o MIME Type desta. O servidor responde o endereço URI que deve ser utilizado para referenciar o recurso criado.

Os MIME Types de conteúdos têm de ser estruturados seguindo a forma: "application/restms+xml", "application/restms+xml", or "text/xml". Caso assim não aconteça o recurso será reconhecimento como uma mensagem.

Atributos adicionais sobre o recurso devem ser especificados nas mensagens que os referenciam.

As mensagens são criadas após todos os recursos, que esta utiliza, estarem criados. A mensagem será enviada para o URI da feed e deverá conter os recursos que deseje embebidos ou referenciados, tendo de ser primeiro criados da forma que foi descrita anteriormente.

Forma de uma mensagem:

<?xml version="1.0"?>

<restms xmlns="http://www.restms.org/schema/restms">

<message

[ address="{address literal}" ]

[ message\_id = "{identifier}" ]

[ reply\_to="{address literal}" ]

[ <header name="{header name}" value="{header value}" /> ] ...

[ <content href="{content URI}" ... />

| <content

type="{MIME type}"

encoding="{encoding}">{content value}</content> ]

</message> ...

</restms>

Semântica dos campos da mensagem:

The address is used in routing, and has feed-specific semantics.

The headers may used in routing, and to carry arbitrary information.

The message\_id is an application-assigned identifier string.

The reply\_to tells an eventual recipient where to send a reply message.

The content hrefs refer to previously staged contents.

The content elements, if they have an element value, hold embedded contents.

Implementations may add other properties with implementation-defined meaning.

Os conteúdos embebidos são codificados em texto claro ou em base64.

As mensagens e os seus conteúdos são enviados para um ou mais pipes e os recursos são automaticamente eliminados.

Uma mensagem não referenciar conteúdos já referenciados, o código de retorno para esta situação será 404 Not Found. Uma mensagem não deve referenciar recursos que foram criados para outra feed, o código de retorno para esta situação será 403 Forbidden. Em situação de erro a mensagem não será encaminhada para os respectivos pipes. O servidor deverá verificar os URI dos conteúdos referenciados numa mensagem, antes de a encaminhar.

Operações autorizadas sobre recursos:

GET - retrieve the content.

DELETE - deletes the content.

A recepção:

A recepção de mensagens é processada da seguinte forma:

- Cliente obtém a mensagem

- Cliente obtém recursos da mensagem

- Cliente apaga a mensagem

Forma das mensagens recebidas:

<?xml version="1.0"?>

<restms xmlns="http://www.restms.org/schema/restms">

<message

[ address="{address literal}" ]

[ message\_id = "{identifier}" ]

[ reply\_to="{address literal}" ]

[ feed="{originating feed URI}" ] >

[ <header name="{header name}" value="{header value}" /> ] ...

[ <content href="{content URI}" ... />

| <content type="{MIME type} encoding="{encoding}">{content value}</content> ]

</message>

</restms>

O apagar duma mensagem causa o apagar dos conteúdos desta.

RestMS allows these methods on a message that is held in a pipe:

GET - retrieves the message representation.

DELETE - deletes the message and all older messages from the pipe.

RestMS allows these methods on a content that is part of a delivered message:

GET - retrieve the content.

DELETE - deletes the content.

