Interface ARS <--> GINSC
Visão Geral

1. Introdução

Este documento visa explicar, de forma sucinta, o funcionamento geral da interface de comunicação entre sistemas legados baseados em REMEDY ARS e GINSC.

Os sistemas legados conectam-se ao sistema GINSC para enviar informações relativas ao fluxo de trabalho correspondente ao legado (workflow do legado) e GINSC, sistema de controle de SLA, tem como objetivo analisar e gerenciar os prazos do fluxo de trabalho. Desse modo, GINSC recebe, através da interface ARS <--> GINSC, informações sobre o fluxo de trabalho do legado e devolve prazos atualizados ao mesmo, pela mesma interface.

Essa interface é composta de dois programas, a saber:

- pARS - Realiza a comunicação com *todos* os legados baseados em REMEDY ARS e transfere as informações para PGINSC.
- PGINSC - Ou IF_GINSC, realiza a comunicação com pARS e com o sistema de gerenciamento de SLAs GINSC.

2. PARS

Para configurar corretamente pARS, é necessário entender primeiro o funcionamento geral de comunicação com sistemas ARS e alguns problemas relacionados. Esse funcionamento será explicado nos itens a seguir.

1. Servidor de notificações

Sistemas baseados em ARS precisam de uma forma para notificarem a interface (no caso, pARS) quando determinados eventos ocorrem. A forma usada para notificar pARS é o servidor de notificações ARS, também chamado de notifier.

Quando iniciamos pARS, o mesmo lê seus arquivos de configuração e se *registra* em um ou mais servidores de notificação. Os servidores de notificação recebem notificações dos legados ARS e armazenam essas notificações em uma fila. Sempre que há um ou mais clientes *registrados* no servidor de notificações, o servidor de notificações abre uma conexão socket com o(s) cliente(s) e envia as notificações recebidas dos sistemas legados.

Essas notificações informam aos clientes registrados quais eventos ocorreram. No caso de nossa interface, os eventos normalmente são criações de novos registros ou alterações de registros já existentes. A notificação também contém informações relativas a esses eventos. No caso de nossa interface, as informações importantes são:

- Qual o registro que sofreu o evento (ID)
- Em qual servidor ARS o registro se encontra

- Em qual esquema (form) dentro do servidor se encontra o registro.

PARS deve ser configurado de forma a se conectar a todos os servidores de notificação necessários para receber os eventos em registros dos legados. Uma vez recebidas as notificações, pARS deve:

- Conectar-se ao servidor de sistema ARS e requisitar todos os dados relativos a notificação;
- Enfileirar os dados para que sejam enviados para pGINSC;
- Enviar os dados a pGINSC;
- Receber a resposta de pGINSC;
- Enviar a resposta de pGINSC de volta ao servidor de sistemas ARS.

Somente depois de receber os dados relativos a notificação corretamente e ter certeza que os dados estão seguros, informar ao servidor de notificações que a notificação já foi recebida e que a notificação pode ser apagada da fila. Perceba que podem haver erros indesejáveis (perda de dados) se mais de um programa, além de pARS, conectar-se ao mesmo servidor de notificações. Essa limitação se deve ao funcionamento do sistema de notificações de ARS.

Uma atenção especial deve ser dada ao segundo item acima. O mesmo será explicado a seguir.

1. **Filas de saída e de erro**

pARS deve enviar dados recebidos de ARS para pGINSC e dados recebidos de pGINSC para ARS. A princípio, parece uma tarefa simples, mas pode ser bastante complicada quando pensamos nos problemas que podem ocorrer no meio do caminho. Vamos começar dos problemas básicos aos complicados.

O problema mais básico que pode ocorrer é pARS receber a notificação do notifier, requisitar com sucesso os dados do servidor ARS e deixar que o notifier apague a notificação da fila, mas obter algum tipo de erro ao enviar os dados coletados para pGINSC. A atitude que pARS toma nesse caso é enfileirar os dados a serem enviados para pGINSC em disco, antes de enviá-los. Dessa forma, não há perda de dados se houver algum problema de comunicação com pGINSC.

Nos casos em que há erro de comunicação, contudo, pARS deve saber tratar o erro. Pode se tratar de um erro temporário, como comunicação de rede, por exemplo, ou de um erro permanente, devido a uma falha grave ou caso não previsto no momento de construção da interface.

PARS deixa então a primitiva (os dados do registro a serem enviados para pGINSC) na fila de saída até que receba uma resposta positiva de pGINSC, indicando que tudo ocorreu ok. Caso não receba uma resposta válida por falha temporária no envio, a mensagem continua na fila para envio posterior. Caso receba uma resposta invalidando a primitiva ou caso algum erro de gravidade semelhante ocorra, a primitiva é então movida para a fila de erros, para que o caso seja tratado manualmente pelos responsáveis.

Nesse momento, um outro possível erro ainda tem de ser evitado. Imaginemos que a primitiva recebida corresponda à criação do registro XXX. Imaginemos agora que a pGINSC retornou um erro grave como resposta do envio dessa primitiva e que, momentos depois, uma primitiva de alteração do registro XXX seja criada em pARS, devido a uma notificação. PARS não pode permitir que essa primitiva seja enviada para pGINSC, uma vez que existe a possibilidade de que o movimento de um registro seja gravado antes de sua criação.

Dessa forma, sempre que uma primitiva estiver pronta para ser enviado, pARS primeiro verifica da fila de erros se existem outras primitivas correspondentes aos mesmo registro. Caso existam, a primitiva é então também movida para a fila de erros, ao invés de ser transmitida para pGINSC.

Imaginemos agora o caso em que tudo ocorreu ok e que uma resposta positiva foi obtida de pGINSC para ser que dados sejam escritos em ARS. Bem, todos os possíveis erros discutidos acima podem também ocorrer na hora de enviarmos a primitiva de resposta para ARS. PARS trata o caso da mesma forma, tendo também uma fila de saída e outra de erros para envios para o servidor de sistemas de ARS.

2. Configuração geral

pARS usa arquivos de configuração em formato XML. O primeiro arquivo é o de configuração de parâmetros gerais e um exemplo é mostrado a seguir.

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<config>
        <!--Variáveis de conexão ao servidor de notificações-->
        <ars_notifier>
                <host>telesp.hi.inet</host>
                <user>GINSC</user>
                <password>GINSC</password>
        </ars_notifier>
        <ars_notifier>
                <host>bauru.hi.inet</host>
                <user>GINSC</user>
                <password>GINSC</password>
        </ars_notifier>
        <ars_system>
                <user>GINSC</user>
                <password>GINSC</password>
        </ars_system>
        <!--Definições para comunicação entre pddr e pginsc-->
        <pginsc_url>http://10.11.29.89:8080/interfaz2/Pginsc</pginsc_url>
        <!--Arquivo de log-->
        <general_log_filename>/tmp/if_ddr_ginscNT.log</general_log_filename>
        <cfg_primitives_folder>/home/mvalle/ARS_IF_GINSC/pcfg</cfg_primitives_folder>
      <pginsc_output_folder>/home/mvalle/ARS_IF_GINSC/poutput</pginsc_output_folder>
        <pginsc_error_folder>/home/mvalle/ARS_IF_GINSC/perrors</pginsc_error_folder>
        <ars_output_folder>/home/mvalle/ARS_IF_GINSC/aoutput</ars_output_folder>
        <ars_error_folder>/home/mvalle/ARS_IF_GINSC/aerrors</ars_error_folder>
        <!-- Outras configurações -->
        <datetime_format>%d/%m/%y</datetime_format> <!-- How to format DATETIMEs -->
</config>
```

As tags ars_notifier definem em quais servidores de notificação pARS deve se conectar. Para isso é necessário informar o host, nome de usuário e senha. Uma vez que pARS receber uma notificação do servidor já saberá em qual host ars, qual schema e entry id deverá busca no servidor de sistema para busca os dados. Faltarão, contudo, o nome de usuário e senha, que devem ser configurados na tag ars_system.

pginsc_url define como realizar a chamada SOAP de pARS para pGINSC. general_log_filename define o nome do arquivo geral de log, onde serão gravadas informações sobre tentativas de conexão, envio de dados, etc. Caso não se queira gravar esse log, é possível configurar a tag para /dev/null.

cfg_primitives_folder define o diretório onde estarão as configurações de primitivas, configuradas para cada legado que se comunicará com pARS. Essa configuração será melhor descrita a seguir. pginsc_output_folder, pginsc_error_folder, ars_output_folder e ars_error_folder definem as pastas que serão usadas para armazenar as filas de saída e de erro, conforme descrito no item anterior. Nessas pastas, as primitivas de um mesmo registro são salvas de forma que se ordenadas em ordem alfabética, estarão também em ordem de criação.

datetime_format define o formato padrão para formatar datas.

3. Configuração para cada legado

Cada tipo de primitiva gerada por pars para ser enviada a pGINSC deve ser configurada primeiro em seu arquivo de configuração correspondente. E como funciona essa configuração? Bem, além dos dados de informação sobre o registro que pARS recebe na notificação, é recebido também, em cada notificação, um número, chamado de ID DE NOTIFICAÇÃO, ou simplesmente o tipo de notificação. Esse número é configurado pelo desenvolvedor do sistema legado e um id de notificação diferente deve ser enviado para cada sistema legado e para cada tipo de evento de cada legado.

Por exemplo, o legado DDR envia código 10 para criação de registro, 11 para alteração. Já o legado SGOS enviaria 12 para criação e 13 para alteração.

Cada arquivo de configuração de primitiva contido em cfg_primitives_folder deverá, então, informar qual é o id de notificação a monitorar. Esse id de notificação se encontra na chave // primitive/legacy/NotifyID do arquivo de configuração da primitiva. Vejamos o exemplo abaixo:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<primitive>
	<!-- Determina o nome de arquivo da primitiva, necessário ter as entradas ID e TIMESTAMP, indicando os parâmetros necessários de cada campo para buscar os mesmos em ARS -->
	<ident>
		<ID data_type="field" data_value="1"></ID>
		<TIMESTAMP data_type="timestamp" data_value=""></TIMESTAMP>
	</ident>
	<!-- configurações de comunicação com o legado-->
	<legacy>
		<!-- ID de notificação a ser monitorado. Sempre que pARS receber uma notificação com esse código, irá usar esse arquivo para criar e transmitir a primitiva. -->
		<NotifyID>10</NotifyID>
		<!-- Host de sistema ARS (não o servidor de notificação). Será substituido pelo nome do servidor recebido pela notificação. -->
		<host>telesp.hi.inet</host>
		<!-- Schema (form) dentro do host, no sistema ARS. Será substituido pelo nome do servidor recebido pela notificação. -->
		<schema>Solicitacao de servico</schema>
		<user>GINSC</user>		<!-- Usuário - ARS System -->
		<password>GINSC</password>		<!-- Senha - ARS System  -->
	</legacy>
	<!-- os dados em si, de requisição -->
	<request PLNAME="PL_SOLICITUDES_INI">
		<!-- On field type nodes, the value means the ARS ID -->
		<PNUM_SOLICITUD data_type="field" data_value="1" field_type="string"></PNUM_SOLICITUD>
		<PTIPO_SOLICITUD data_type="field" data_value="600000075"  field_type="string"></PTIPO_SOLICITUD>

		<PRESPONSABLE data_type="field" data_value="4" field_type="string"></PRESPONSABLE>
		<PRAZON_SOCIAL data_type="field" data_value="536870939" field_type="string"></PRAZON_SOCIAL>
		<PCNPJ data_type="field" data_value="600000076" field_type="string"></PCNPJ>
		<PFECHA_ASIGNACION data_type="field" data_value="600000021" field_type="datetime"></PFECHA_ASIGNACION>
		<PFECHA_SOLICITUD data_type="field" data_value="540000200" field_type="datetime"></PFECHA_SOLICITUD>
		<PFECHA_CREACION data_type="field" data_value="3" field_type="datetime"></PFECHA_CREACION>
		<!-- O formato de dados a seguir (valor quando o tipo de dado é timestamp) está documentado na man page da função strftime (digitar “man strftime”, no console)  -->
		<PTIMESTAMP data_type="timestamp" data_value="%d/%m/%Y %H:%M:%S"></PTIMESTAMP>
		<PSISTEMA data_type="value" data_value="SGDDR"></PSISTEMA>
		<PSERVICIO data_type="value" data_value="DDR"></PSERVICIO>
		<PESTADOS>
			<PEDIDO data_type="field" data_value="600000028" field_type="string"></PEDIDO>
			<JUNTOR data_type="field" data_value="536871155" field_type="string"></JUNTOR>
			<MXT data_type="field" data_value="536871154" field_type="string"></MXT>
		</PESTADOS>
		<PSALIDA data_type="output"></PSALIDA>
	</request>
	<!-- Como pARS espera a resposta. PGINSC irá comparar o nome da tag com o nome de cada valor que ele receber como resposta do módulo de regras do GINSC  -->
	<response>
		<!-- campo Data Entr. Prog.1   600000041 -->
		<UMBRAL data_type="field" data_value="600000041" field_type="datetime"> pginsc preenchera aqui com a resposta </UMBRAL>
	</response>
	<!-- Determine os parâmetros de notificação a GINSC, para que pGINSC possa chamar  o servlet de notificação em GINSC -->
	<NOTIF>
		<URL>http://localhost:8080/IF_GINSC/test_dom2</URL>
		<USER>GINSC</USER>
		<PASSWORD>GINSC</PASSWORD>
		<QUERY>
			<SISTEMA data_type="value" data_value="SGDDR"></SISTEMA>
			<SERVICIO data_type="value" data_value="DDR"></SERVICIO>
			<SOLICITUD data_type="field" data_value="1" field_type="string"></SOLICITUD>
			<TIPOSOLICITUD data_type="field" data_value="600000075"  field_type="string"></TIPOSOLICITUD>
		</QUERY>
```

```
        </NOTIF>
</primitive>
```

No exemplo acima, ident define a identificação da primitiva. No caso, a primitiva é identificada pelo EntryID do registro e pelo timestamp recebido durante a notificação. O nome de arquivo é também formado a partir dessa identificação. Perceba que os nós dentro de ident tem alguns atributos como data_type e data_value. Esses atributos servem para identificar a origem do campo. Todos os campos contidos em //primitive/NOTIF/QUERY, //primitive/ident, //primitive/request e //primitive/response funcionam da seguinte forma:

- Caso o atributo data_type seja 'timestamp', o valor do campo será substituido pelo timestamp recebido na notificação e o atributo data_value indicará como o timestamp deverá ser formatado.
- Caso o atributo data_type seja 'output', pARS simplesmente o ignora.
- Caso o atribute data_type seja 'field', pARS irá buscar ou escrever o valor, dependendo do caso, em ARS. data_value será o ID do campo em ARS e field_type será o tipo de dado dentro do ARS. Os tipos de dados suportados são string, integer, datetime e real.

//primitive/legacy contém os parâmetros de configuração de comunicação com o sistema legado. Alguns desses parâmetros são substituidos na hora em que pARS recebe a notificação. Esses parâmetros indicam como escrever a primitiva no servidor de sistema ARS.

//primitive/request contém os dados em si da primitiva, ou seja, do registro a ser enviado para pGINSC. PARS substitui os valores dos campos de acordo com o que está especificado nos atributos e envia a primitiva para pGINSC. Um atributo importante é o nome da PL a ser chamada por pGINSC para que os dados sejam escritos.

//primitive/response informa a pGINSC como esperamos os dados de resposta. Ao receber a primitiva de pARS, pGINSC fará o parse do XML e enviará os valores (como pares nome/valor) para GINSC. Depois disso, pGINSC recebe os dados de resposta, também como pares nome/valor. Para cada par nome/valor, pGINSC compara o nome do campo recebido na resposta com cada campo recebido em //primitive/response. Se o nome bater, o valor é substituido na primitiva, para que a mesma seja enviada de volta a pARS. Observe que pARS já saberá como escrever os dados em ARS, uma vez que os parâmetros de descrição de campo continuam inalterados.

//primitive/NOTIF contém todas as informações adicionais necessárias para que pGINSC consiga escrever e ler dados de GINSC. Isso inclui a url de comunicação com o módulo de regras e alguns dados (em REQUEST) que são provenientes do registro ARS.

3. PGINSC

Uma vez que a primitiva foi enviada de pARS para pGINSC com toda segurança, pGINSC substitui valores na primitiva recebida e a envia de volta para pARS. Um exemplo de primitiva vinda de pGINSC para pARS é mostrado a seguir:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<primitive>
        <ident>
                <ID data_type="field" data_value="1"></ID>
                <TIMESTAMP data_type="timestamp" data_value=""></TIMESTAMP>
        </ident>
        <legacy>
                <NotifyID>10</NotifyID>
                <host>telesp.hi.inet</host>
                <schema>Solicitacao de servico</schema>
                <user>GINSC</user>
                <password>GINSC</password>
        </legacy>
        <response>
                <UMBRAL data_type="field" data_value="600000041" field_type="datetime">20/05/2004 12:13:25</UMBRAL>
        </response>
        <NOTIF>
                <URL>http://localhost:8080/IF_GINSC/test_dom2</URL>
```

```
                <USER>GINSC</USER>
                <PASSWORD>GINSC</PASSWORD>
                <QUERY>
                        <SISTEMA data_type="value" data_value="SGDDR"></SISTEMA>
                        <SERVICIO data_type="value" data_value="DDR"></SERVICIO>
                        <SOLICITUD data_type="field" data_value="1" field_type="string"></SOLICITUD>
                        <TIPOSOLICITUD data_type="field" data_value="600000075"  field_type="string"></TIPOSOLICITUD>
                </QUERY>
        </NOTIF>
        <status>
                <id>Ok</id>
                <message>Everything has gone fine...</message>
        </status>
</primitive>
```

Como se pode perceber, quase nada é mudado na primitiva. Os dados de request são retirados, por não serem mais necessários em nenhuma etapa subsequente do processo. Os campos em response tem seu valor preenchido de acordo com a resposta obtida de GINSC. É adicionado um nó adicional chamado status, com um identificador e uma mensagem. O campo id indica o código de erro e vale 'ok' em caso de sucesso em todas as etapas. Em caso de erro, id indica qual foi o erro ocorrido e message indica uma mensagem verbal a ser colocada no log de pARS.

4. Tecnologias utilizadas

Esta seção descreve alguns detalhes de quais tecnologias foram utilizadas na implementação do projeto que podem ser relevantes para obter sucesso na instalação, configuração e manutenção do mesmo.

PGINSC é feito em Java e se comunica com pARS via uma api SOAP para java chamada SAAJ. A versão do JSDK usada para a criação do projeto foi a 1.4. pGINSC usa uma api comum do sistema GINSC (um arquivo jar) que é usado na comunicação de pGINSC com GINSC. pGINSC é o módulo de interface do sistema GINSC e deve rodar junto com o mesmo em um container para servlets java (no desenvolvimento foi usado o tomcat versão 5.0). Foi utilizado o IDE Eclipse (http://www.eclipse.org) no desenvolvimento.

pARS é feito em ANSI C e é linkado com algumas librarias, a saber:

- libXML2 versão 2.6.9. Pode ser encontrada em http://www.xmlsoft.org. Essa versão da lib deve ser entregue junto com o projeto, para que não hajam problemas de versionamento.
- libARS4.0. Trata-se da API ARS distribuida junto com REMEDY ARS. Foi utilizada a versão 4.0, encontrada no ambiente de desenvolvimento. São utilizadas tanto a lib de sistema (libar.a) como a de notificação (libnts.a). O Makefile de ARS oferece maiores detalhes.
- pthread, socket, e outras libs que normalmente já se encontram instaladas nos ambientes UNIX (em especial, em Solaris, ambiente utilizado no desenvolvimento e nos testes).
- libcsoap. Trata-se de uma lib open source que se encontrava em desenvolvimento no momento em que o projeto era desenvolvido. Embora em desnvolvimento, tratava-se da melhor API C para utilização de SOAP no momento. Por esse motivo e por ser open source, foi utilizada no desenvolvimento. **O código da lib foi alterado**, de forma que **se torna de extrema importância usar a lib entregue junto com o pacote**, pelo fato de que até o término do projeto ainda não haviam sido lançada uma versão da lib que incluisse os patches (modificações) criadas pelo nosso desenvolvimento. A página oficial do projeto é http://csoap.sf.net.