Sistema Integrado de Administración Tributaria
SIAT

Arquitectura SIAT
Anexo I – Administrador de Procesos

Abril de 2007

Contenido

Administrador de Procesos - ADP

Versiones

Fecha	Version	Descripcion	Autor
01/04/07	1.0	Inicial	tecso:
29/10/07	1.1	Entrega Diseño	tecso:

Objetivos del documento

Describir la arquitectura y diseño del Administrador de Procesos (ADP) del SIAT.

Introducción

Entendemos por Arquitectura del Software a la organización fundamental de un sistema en términos de: sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. (IEEE Std 1471-2000)

En este marco, la definición de la arquitectura deberá establecer aspectos esenciales para el desarrollo del proyecto de acuerdo a tres vistas:

- Vista Lógica
- Vista Física
- Vista de Desarrollo

Objetivos del Administrador de Procesos

A continuación, las enunciamos:

- Ejecutar procesos con tiempos de ejecución prolongados.
- Iniciar las ejecuciones por Demanda, Programados, o Por arribo de archivos.
- Ejecutar procesos de forma Asíncrona aislada de la sesión web de SIAT.
- Ejecutar procesos en un entorno aislado.
- Permitir ejecutar procesos en varios pasos.
- Control de accesos por ejecución utilizando SWE.
- Permitir registrar las actividades de los procesos durante su ejecución.
- Presentar una interfaz web para administrar los procesos y corridas.
- Exponer un API para ejecutar y monitorear procesos de forma externa.
- Facil y Rapida administracion y programación de nuevos procesos.
- Posiblidad de ejecutar procesos en Java o delegar el procesamiento a Store Procedures.

Premisas

Para el diseño y programación del Administrador de Procesos se tienen en cuenta las seguientes premisas:

- Su diseño y arquitectura no debe alejarse de sus reales objetivos de implementar sus funcionalidades.
- Diseño Simple: ADP se mantiene simple haciendo foco en un conjunto pequeño de requerimientos.
- El uso de ADP debe ser sencillo.
- ADP confía lo más que puede en la correctitud de los procesos que ejecuta.
- Al mismo tiempo provee mecanismos a prueba de fallos.
- Utiliza la Base de Datos de forma segura para mantener coherente el estado de sus procesos.

Vista lógica

Descripción general

Es util distinguir los siguientes conceptos en el Administrador de Procesos (ADP):

- Procesos
- Programas de un Proceso
- Parámetros de Procesos
- Corridas de procesos y Estados
- Valores de Parámetros de Corrida
- Log de Corrida
- Consola de Administración Web
- Seguridad
- ADP API

Procesos

Con Proceso nos referimos a la definición de la forma y modo de ejecución de un programa. Por ejemplo un Proceso contiene información que lo define como, Nombre, Tipo de Programa que ejecuta (Java o SP informix), Modo de ejecución (Sincrónico o Asíncrono), etc..

En detalle un Proceso lo definen:

- **Nombre y Descripción de Proceso**:
  Nombre único que se utilizara para identificarlo desde la programación al momento de ejecutarlo u obtenerlo. La descripción es una cadena descriptiva del proceso.

- **Tipo de Programa:**
  Indica el tipo y programa que llevará adelante el proceso. ADP soporta Java y Store Procedures como tipos de programa a ejecutar. Esto significa que la lógica

del proceso podrá ser representada en Clases Java o en Store Procedures almacenados en Base de Datos.

Para el caso de Clases Java se debe informar a ADP el nombre de la clase que realiza la lógica del Proceso. Dicha clase deberá respetar una interfaz bien definida por ADP.

Para el caso de Store Procedures se debe informar el nombre de los SP en la Base de datos.

- **Modo de ejecución:**
  Indica el comportamiento de ADP al momento de recibir la orden de iniciar la ejecución de un programa.
  El modo de ejecución puede ser Sincrónico o Asíncrono.

- **Tipo de Inicio de ejecución:**
  Indica la forma en que es iniciado el programa. ADP soporta dos formas de Inicio: **Manual** o **Por arribo de Archivo**.

  La forma Manual, requiere la intervención explicita para ejecutar el proceso. El usuario o SIAT podrá elegir si comenzar la ejecución Inmediatamente o Programarla para una determinada fecha.

  En la forma Por Arribo de Archivo, ADP comienza la ejecución del proceso de forma Inmediata al llegar o cambiar un archivo de un directorio previamente definido para el proceso.

  (*ver Forma de Inicio de Corrida*)

- **Directorio Input:**
  Indica que directorio observar para el arribo de archivo para comenzar la ejecución del proceso.

- **Parámetros de Proceso:**
  Indican un conjunto de atributos a valorizar antes de la ejecución de un proceso

Programas de Proceso

Con programa de un Proceso, nos referimos al fragmento de código que lleva a cabo la logica de un proceso.

ADP permite asociar cuatro programas a cada Proceso, dichos programas son:

- El programa Execute
  Este programa representa la lógica del proceso que se ejecuta cuando ADP inicia la Corrida de un Proceso que esta en estado Espera Comenzar. *(ver Corridas de Procesos y Estados)*
- El Programa Resume
  Este programa representa la lógica del proceso que se ejecuta cuando ADP inicia la Corrida de un Proceso que esta en estado Espera Continuar. *(ver Modelo de procesos)*

- El Programa Validate:
  Es un programa que retorna Verdadero o Falso. ADP invoca a este programa justo antes de Invocar a Execute o Resume.
- El Programa Cancel
  Este programa representa la lógica que se ejecuta cuando el usuario selecciona Cancelar una Corrida.

Para el caso de un Proceso con Tipo de Programa Java, cada programa corresponde a un método de una clase a implementar que respeta una interfaz de ADP.

Para el caso de un Proceso con Tipo de Programa Store Procedure cada programa corresponde a un nombre de SP almacenado en la DB del SIAT.

Ya sea el nombre de la clase java o los nombres de los SPs, estos datos son ingresados al momento de dar de alta un Proceso en ADP junto con sus datos y sus parametros.

Parámetros de Proceso

Indican un conjunto de atributos a valorizar antes de la ejecución de un proceso. Estos atributos deben interpretarse como los parámetros de un programa o método. Por ejemplo, "Zona a Emitir" podría ser un parámetro del proceso de Emisión TGI o "Id Selección Almacenada" podría ser un parámetro del proceso de envío de deuda a judicial.

Un parámetro de proceso esta asociado con un atributo de las tablas de definición de SIAT. Ademas un Parámetro puede ser marcado como InputFile, lo que indica que el valor de este parámetro contendrá el path a un archivo que se utilizará como input del proceso.

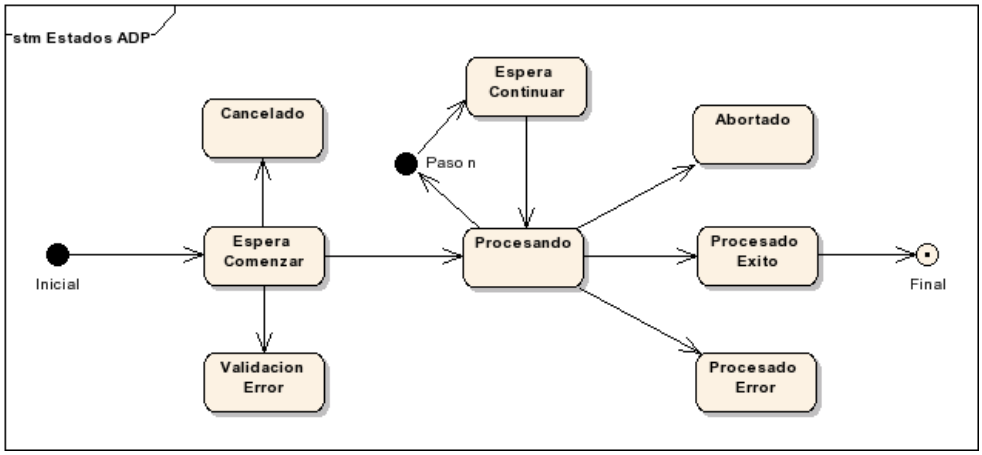
Corridas de Proceso y Estados

Definimos Corrida, a la instancia de un Proceso. Una corrida representa una instancia de un proceso con sus parámetros valorizados.

Por otro lado una Corrida posee estados que indican, si el programa del proceso esta siendo ejecutado, si ya termino, o si esta por ejecutarse. También existen estados que indican la forma en que se ejecuto el programa del proceso.

Ademas una Corrida posee un Mensaje de Observación que se utiliza para brindar información a nivel usuario del Estado actual de la Corrida.

La relación entre los posibles estados de ADP, se muestran en el diagrama:



Descripción de los Estados:

- **En espera comenzar**
  Representa una corrida que esta programada para ejecutarse, el comienzo puede ser Inmediato o a una determinada fecha y hora. (*ver Formas de inicio de Corrida*)

- **Procesando**
  ADP, inicio el programa relacionado con el Proceso, y se esta ejecutando.

- **Procesado con éxito**
  El programa termino con éxito y finalizo el proceso.

- **En espera continuar**
  El programa termino con éxito, pero al finalizar indico que todavía quedan pasos pendientes para finalizar el proceso. (*ver Procesos de varios pasos con estados intermedios*)

- **Procesado con error**
  El programa termino con un error

- **Validación con error**
  El programa no pudo comenzar porque no paso la validación. (*ver Validación durante el inicio*)

- **Cancelado**
  El usuario Cancelo el proceso.
  Importante: un proceso puede pasar a estado cancelado solo si se encuentra en uno de los estado: En espera comenzar, En espera continuar, Procesado con error, Validación con error.

- **Abortado por excepción**

Representa una terminación anormal de la ejecución del programa del proceso. Esta forma de terminación puede suceder cuando:

- El programa Java lanza una excepción que no es manejada correctamente
- Se interrumpe la conexión a la DB, Se agota la memoria en el VM, etc.
- Cuando el contenedor de ADP (Tomcat, etc), es finalizado mientras existen corridas en estado 'Procesando'
- etc...

Valores de Parámetros en las corridas

Cuando SIAT invoca a ADP para ejecutar un proceso, le envía los valores de los parámetros del Proceso. Estos valores son almacenados en tablas de ADP y quedan relacionados con una corrida a través de un ID de Corrida, dicho ID es informado a SIAT para que lo almacene.

Estos valores almacenados en las tablas de ADP tiene dos usos fundamentales.

Envió de Valores a los programas
Cada vez que sADP debe comenzar el inicio de un programa, hace disponible estos valores al programa.

Para el caso de clases Java, los valores son enviados en forma de tabla clave, valor accesibles desde el método java que se va a ejecutar.

Para el caso de los procesos tipo Store Procedure, el Id de Corrida es enviado como un parámetro del SP. Con este único dato, los SP pueden acceder a los parámetros valorizados de esa corrida.

Uso desde SIAT de los valores
Por otro lado SIAT podrá brindar información sobre el estado y valores de Corrida de sus procesos utilizando el ID de Corrida almacenado y consultando las tablas bien definidas de ADP.

Log de Corrida

Ademas del logger especifico e interno que ADP utiliza para registrar sus tareas y funcionamiento, ADP brinda un servicio de Log por Corrida para los Procesos Java.

ADP genera un archivo de log asociado a cada Corrida, y expondrá servicios para agregar información de log a este archivo. Es responsabilidad del programa del proceso utilizar este servicio adecuadamente para registrar datos útiles.

Consola de administración Web

Esta es una Interfaz gráfica Web de Acceso restringido por roles. Desde esta interfaz se puede consultar los resultados y estados de los procesos ejecutados y ejecutándose.
También permite modificar y dar de alta Procesos con sus parámetros de ejecución. Así como todas las tareas pertinentes a la administración de los procesos.

Seguridad

ADP controla el acceso a la ejecución de cada procesos, para cada usuario.

ADP utiliza el esquema de roles de SWE para controlar el acceso a sus funcionalidades. Para ejecutar un proceso, requiere las credenciales del usuario que realiza la acción, y verifica en SWE que posea permisos para crear y/o obtener el proceso especifico.

API de ADP

Para facilitar y estandarizar la programación de procesos, ADP provee una pequeña API para llevar a cabo las tareas más cotidianas durante la programación de un Proceso.

Las API de ADP contemplan:

- Lectura/Escritura de archivos de texto plano:
  Archivos formateados en columnas, frecuentemente usados por archivos de input/output

- Manipulación de Archivos del Proceso:
  Mover/Borrar/Renombrar los archivos de input del proceso

- Lectura/Escritura de Estado de Corrida:
  Modificación y lectura de Estado, Paso, Mensaje de observación del Estado y otros

- Lectura/Escritura de Valores de Parámetros de Corrida
  Acceso a los valores de parámetros almacenado en la corrida.

- Ejecución de StoreProcedures
  Helper de para ejecutar Store Procedures vía JDBC

Modelos de ejecución de los procesos

Formas de inicio de los programas

Con inicio de programa, nos referimos a la forma en que comenzara la ejecución del programa del Proceso, osea, a la transición de la corrida al estado **Procesando**.

ADP inicia el programa según estas tres maneras:
- Manual Inmediata
- Manual Programada
- Por Arribo Archivo

Las primeras dos formas corresponden a Procesos con tipo ejecución Manual. La ultima corresponde al tipo ejecución Por Arribo Archivo.

Para el tipo ejecución **Manual** el usuario, a través de la aplicación SIAT, elige comenzar un proceso, para ello ingresa los parámetros del proceso, y elige como ejecutarlo, si bien Inmediatamente o bien elige una fecha y hora a la cual se iniciará (forma Programada).

Ya sea Inmediatamente o de forma Programada ADP crea la Corrida y la coloca en estado En Espera Ejecución, luego ADP detecta que hay procesos en Espera y si debe comenzar la ejecución de algún programa ya sea porque llego su hora programada o porque debe iniciarse inmediatamente.

Para el tipo de ejecución **Por Arribo de Archivo,** ADP contiene un monitor que revisa periódicamente los directorios de los procesos de este tipo. Cuando el monitor detecta un nuevo archivo o un cambio en la fecha de modificación de algún archivo, llama a ADP creando una Corrida de forma Inmediata y asyncronicamente. El monitor pasa a esta Corrida el nombre del archivo que arribo, como valor de un parámetro del proceso.

Validaciones durante el inicio

Previo al inicio del programa que realiza la lógica del proceso, ADP hace una llamada al programa de Validate asociado al proceso.

Si Validate retorna True, continua con la ejecución del programa Execute o Resume.

Si Validate retorna Falso ADP pasa la Corrida a estado Validate Error.

El programa Validate recibe exactamente los mismos valores de parámetros que reciben los demás programas. Esto permite a Validate realizar la lógica para determinar si están dadas las condiciones para realizar el procesamiento solicitado.

De manera opcional, ADP soporta que cuando se Ejecuta un proceso, ya sea Inmediatamente o de forma Programada, se ignore el paso de validación.

Procesos de varios pasos con estados intermedios

En SIAT se detecto la presencia de procesamientos funcionales que se realizan en varias etapas, o pasos. Por ejemplo, el Proceso de Emisión de TGI se podría dividir en tres o más etapas.

- Cálculo
- Impresión de las boletas.
- Incorporación de la deuda

(**Nota**: Esta división en tres etapas, se pone como ejemplo a fines de ilustración. Y no representa los verdaderos pasos del proceso Emisión TGI)

Para soportar este requerimiento, se opto por hacer que ADP permita múltiples ejecuciones de programas para una única Corrida, y la incorporación del estado Esperando Continuar.

Más en detalle, ADP utiliza el estado Esperando Continuar , el programa Resume y el número de paso como dato de la Corrida.

Para estos casos el comportamiento de ADP consiste simplemente en llamar al programa Resume cuando el estado de la corrida es Esperando Continuar.

Es responsabilidad de los programas que implementan el proceso manejar adecuadamente el pasaje a cada estado. Así y todo ADP controla que la transición de estado sea una transición valida según su diagrama su diagrama de Estado.

Por ejemplo, para el caso anterior la Corrida pasara por los siguientes estados:

1) Etapa 1: Estado Esperando Comenzar (Nro. Paso en Corrida = 0)
2) Se cumple hora programada: Estado Procesando
3) Etapa 2: Esperando Continuar (Nro. Paso en Corrida = 1)
4) Se cumple hora programada: Estado Procesando
5) Etapa 3: Esperando Continuar (Nro. Paso en Corrida = 2)
6) Se cumple hora programada: Estado Procesando
7) Procesado Éxito (Nro. Paso en Corrida = 3)

Nota: en la sección Uso de ADP, se describe en detalle el ejemplo de procesamiento en tres etapas.

Manejo de errores y fallos inesperados

ADP posee dos estados de error diferentes para indicar que ocurrió un error durante la ejecución de un programa de Proceso.
A continuación se describen los dos estados, y las situaciones que llevan a estos estados.

Estado Procesado Error

Representa un Error Lógico o Funcional que ocurre durante el procesamiento de los programas Execute o Resume. Por ejemplo, para el procesamiento de novedades del padrón, supongamos que llega un dato para dar de alta una parcela cuyo radio no esta dentro de los valores permitidos.

Entonces, digamos que el programa detecta este error y que funcionalmente no se puede continuar. En este caso, el programa es responsable de pasar el estado de la Corrida a Procesado Error llamando a las API de ADP, y terminar la ejecución.

Ademas, opcionalmente el programa podrá modificar la Observación de Estado asociada a la Corrida, para brindar información al usuario de la causa del detenimiento, y también podrá utilizar el Log de Corrida para volcar la información de bajo nivel que causo el error.

Pero, de manera desafortunada, puede ocurrir que el programa NO detecte la inconsistencia y se produzca una excepción no manejada. Para estos casos es donde aparece el segundo estado de manejos de error.

El estado Abortado

Podemos distinguir tres causas por la cual se llega a este Estado:

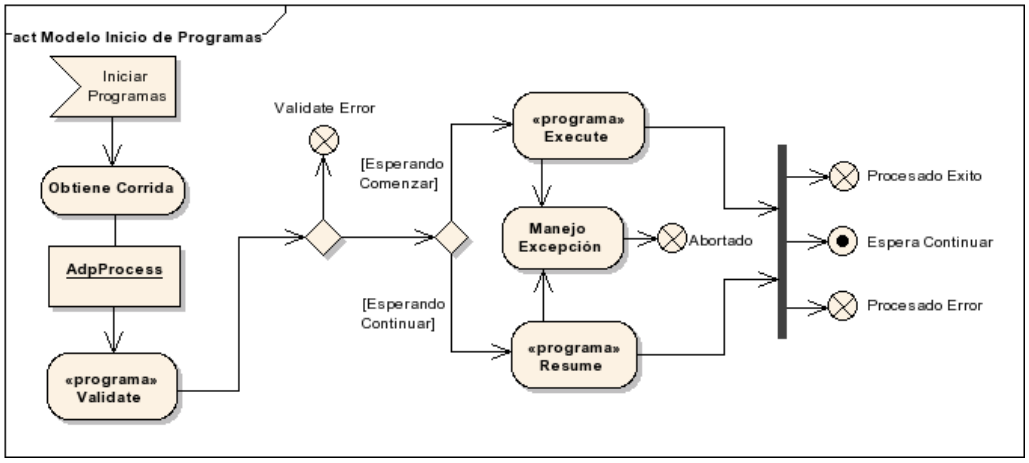
1) La inconsistencia produce una excepción no manejada:
En este caso, la excepcion sube y ADP atrapa la excepción, a estas altura el programa a terminado abruptamente por la excepcion. Entonces, ADP, cambia el estado de la corrida a Abortado, coloca una Observacion de estado estandar, y coloca el detalle de la excepcion en el log de la corrida. Es de esperar que no ocurran errores funcionales por esta via, y que el programa del proceso maneje adecuadamente cada error funcional que pueda ocurrir.

2) Errores causados por acceso a recursos, producen una excepción no manejada:
Puede ocurrir que durante el procesamiento, el intento de utilizar un recurso del sistema falle, por ejemplo, Agotamos la memoria del Virtual Machine, Falla una conexión a la Base de Datos, Falla el acceso a algún archivo por problemas de permisos, etc...
En estos casos, el comportamiento de ADP es similar al caso anterior, ya que ADP no puede distinguir si la excepción se trata de un error funcional o de recursos.

3) Corte abrupto del procesamiento por causas externas:
Puede ocurrir que todo el sistema de ADP se detenga mientras están ejecutándose programas. Esto puede ocurrir o por una falla inesperada en ADP, porque el contexto en Tomcat del ADP finaliza abruptamente, etc, etc.
Sea como sea, desde el punto de vista de la corrida, el procesamiento a terminado de forma inesperada.
En este caso, la próxima vez que se levante ADP, coloca todos los procesos que estaban ejecutándose en estado Abortado, con una observación de estado adecuada. Mientras tanto de forma inevitable, la Corrida aparece en Estado Procesando, pero en realidad no hay ningún programa ejecutándose, ya que ADP esta detenido.

Diagrama de Actividad de Inicio de Programas

El siguiente diagrama ilustra las actividades para de ADP para el inicio y ejecución de los programas de un proceso.



Ejemplo de funcionamiento de ADP

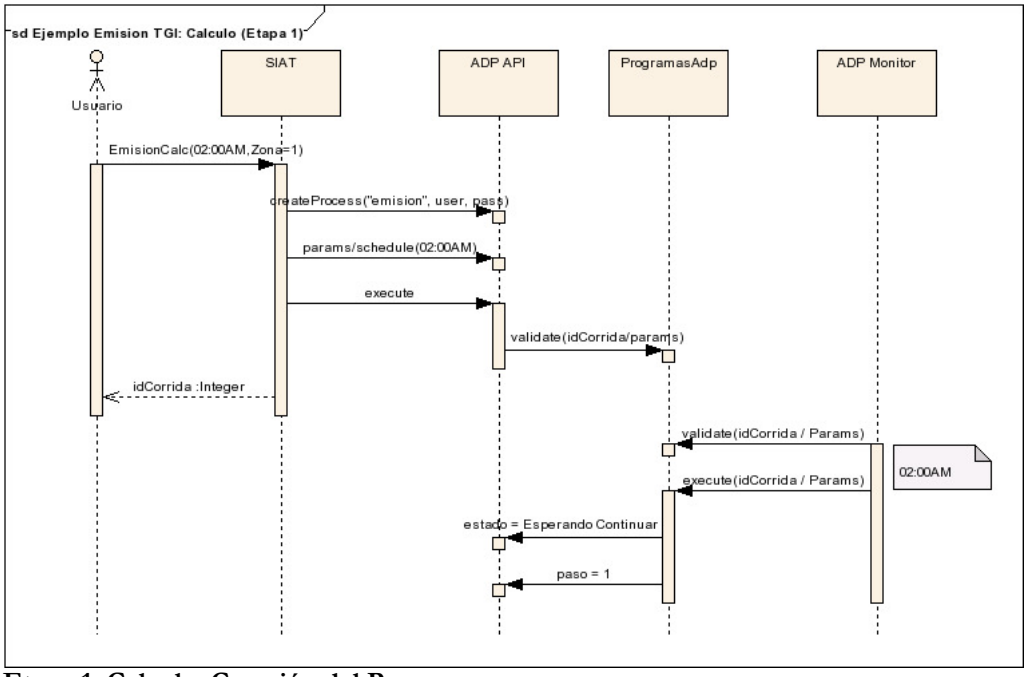
Para ilustrar la interacciones y uso de ADP con SIAT, podemos tomar como ejemplo la división en tres pasos de Emisión, puesta como ejemplo en la sección anterior.

- Cálculo
- Impresión de las boletas.
- Incorporación de la deuda

La siguiente tabla ilustra como sería la realización de un proceso de varios pasos con sus estados intermedios para el ejemplo de arriba:

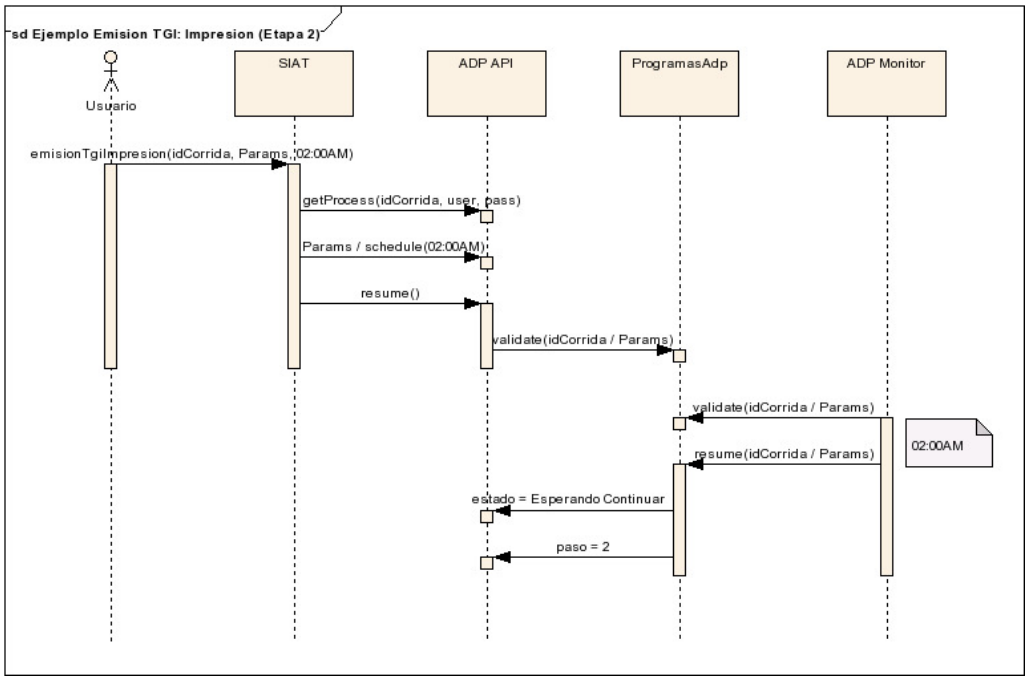
Evento	Actividades
Etapa 1 Calculo - El usuario inicia el proceso de emisión. - Ingresa los parámetros del proceso - Y decide correrlo de forma Programada a las 02:00hs.	Se instancia la Corrida en Estado '**En Espera Comenzar**' ADP retorna el Id de Corrida creado.
Se cumple el horario programado	- Se ejecuta programa Validate - Se pasa a estado Procesando - Se ejecuta programa **Execute** - El programa Execute al finalizar pasa el estado de la Corrida a **En Espera Continuar** y Paso igual 1 La corrida ahora se encuentra en estado En Espera Continuar con Paso igual a 1
Etapa 2 Impresión boletas - El usuario continua con el segundo paso de Impresión de boletas - Programa la corrida para ejecutarse a las 02:00hs.	Se obtiene la corrida Se programa la fecha y hora de ejecución.
Se cumple el horario programado	- Se ejecuta programa Validate - Se pasa a estado Procesando - Se ejecuta programa **Resume** (Aquí Paso tiene el valor 1) - El programa Resume al finalizar pasa el estado de la Corrida a **En Espera Continuar** Paso igual 2 La corrida ahora se encuentra en estado En Espera Continuar con Paso igual a 2
Etapa 3 Incorporación de deuda - El usuario continua con el tercer paso final del proceso. - Programa la corrida para ejecutarse a las 02:00ha.	Se obtiene la corrida Se programa la fecha y hora de ejecución.
Se cumple el horario programado	- Se ejecuta programa Validate - Se pasa a estado Procesando - Se ejecuta nuevamente programa **Resume** (Aquí Paso tiene el valor 2) - El programa Resume al finalizar pasa el estado de la Corrida a **Procesado éxito** y Paso igual 3 El proceso ha concluido. Y la corrida se encuentra en estado Procesado éxito.

A continuación mostramos los tres diagramas de secuencia correspondiente a cada etapa, según el ejmplo de la tabla anterior.

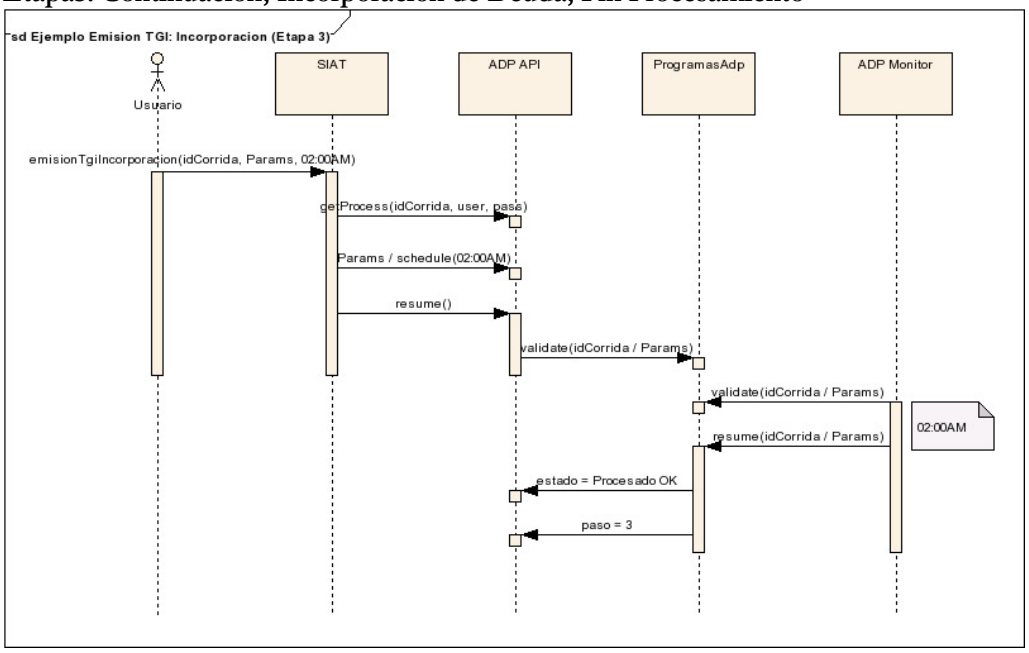


Etapa 1: Calculo, Creación del Proceso.

Etapa 2: Impresión



Etapa3: Continuación, Incorporación de Deuda, Fin Procesamiento



Vista Desarrollo

Descripción general

En esta sección explicaremos cuales son las tareas y pasos para desarrollar un programa de corrida para que pueda ser utilizado por ADP.

Implementación de procesos ADP

En esta sección se explica la manera de implementar un programa para que pueda ser utilizado como un programa de proceso de ADP.

Programas de Proceso tipo Java

Digamos que se desea implementar un proceso que estará programado en java. Para eso, se deben realizar tres tareas:

1) Programación de la clase Java
2) Compilación y Despliegue
3) Alta del Proceso en ADP
4) Alta de la acción en SWE

Para la **programación de la clase Java** se debe construir una clase que implemente la interfaz AdpWorker que posee los siguientes métodos:

```
public void execute(AdpProcess adp) throws Exception;
public void resume(AdpProcess adp) throws Exception;
public boolean validate(AdpProcess adp) throws Exception;
public void cancel(AdpProcess adp) throws Exception;
```

ADP llama a cada método cuando corresponde y pasa la clase AdpProcess instanciada con los valores de la corrida, y sus parámetros. Luego cualquier cambio de estado o lectura/escritura de parámetros, etc... se puede realizar con esta clase.

La clase AdpProcess permite:

- Obtener los parámetros del proceso y sus valores de corrida
- Obtener/Modificar el estado de la Corrida
- Obtener el Id de Corrida y Datos de fechas de inicio, fin, etc.
- Obtener/Modificar el Mensaje de Observación de Estado
- Obtener Log de Corrida.

Para la **compilación y despliegue** de la clase, se ha definido una estructura de directorios especifica para alojar las clases de programas de proceso. En esta estructura de directorios , se cuenta con scripts ant que realizan la compilación y el despliegue.

Disponer de una estructura especifica presenta la ventaja que simplifica el desarrollo y compilación de nuevos procesos, y ademas, separa claramente la ubicación del código fuente de los procesos..

Una vez hecho el despliegue del nuevo proceso, se debe realizar el **Alta de Proceso en ADP.** Para esto se debe ingresar a la consola de ADP, y crear un nuevo proceso, con Tipo Programa

Java e informar el nombre del proceso, descripción, nombre completo de la Clase recién programada, y datos de forma de corrida, junto con los parámetros del Proceso.

El a**lta de la acción en SWE** consiste en dar de alta de la acción (y el rol se fuese necesario) del proceso en SWE para ejecutar el Proceso. El nombre de la acción debe respetar el siguiente nombre: /adp/[nombre proceso]/ejecutar

Programas de Proceso tipo SP

Para crear un nuevo proceso se necesita realizar dos pasos:

1) Programación de los Store Procedures necesarios para el Proceso.
2) Alta del Proceso en ADP
3) Alta de la acción en SWE

Llegado el momento del inicio de corrida, ADP invoca a cada SP cuando corresponde. El SP Validate solo es invocado si corresponde y existe en los datos de definición del proceso.

Cada SP es invocado pasando el Id Corrida como parámetro. Es responsabilidad de cada SP, obtener los parámetros con sus valores y cambiar los datos y estado de la corrida accediendo directamente a las tablas de ADP.

Una vez programado el o los SP del nuevo proceso, se debe realizar el **Alta de Proceso en ADP.** Para esto se debe ingresar a la consola de ADP, y crear un nuevo proceso, con Tipo Programa SP e informar el nombre del proceso, descripción, nombre en la DB de cada SP creado (Execute, Resume, Validate...), y datos de forma de corrida, junto con los parámetros del Proceso.

El a**lta de la acción en SWE** consiste en dar de alta de la acción (y el rol se fuese necesario) del proceso en SWE para ejecutar el Proceso. El nombre de la acción debe respetar el siguiente nombre: /adp/[nombre proceso]/ejecutar

Compilación y estructura de directorios

Respecto al código fuente de los procesos de SIAT y ADP, lo separamos en tres partes:

- Código fuente de la Consola de Administración de ADP
  Es el código que implementa los Mantenedores de los Procesos y de las Corrida, etc. Se programa como un modulo más de SIAT, con accesos controlados por Roles.

- Código fuente de el core de ADP
  Es el código especifico de ADP. Aquí se encuentran las clases monitores de arribo de archivos, y ejecución de los programas, etc. También se encuentran aquí las clases que utiliza SIAT para lanzar e implementar los procesos.

- Código fuente de los procesos SIAT que se ejecutan en ADP
  Es la parte del código de SIAT que se implementa mediante procesos de ADP. Desde el punto de vista lógico, son clases estructuradas en paquetes definidos

con el mismo criterio utilizado para el resto de las funcionalidades de SIAT, salvo que se encuentran separadas en otro directorio.

La siguiente es una tabla que muestra la estructura de directorios, y los binarios generados según las partes enunciadas arriba.

Directorio	Descripción	Nombre del jar

`/iface`
`/buss`
`/view`
	Consola de Administración ADP
	(Y los otros módulos de SIAT)
		`proSiat.jar`
		(y los otros jar por módulo de SIAT)

`/adpcore`	Core de ADP	`adpcore.jar`
`/adpsiat`	Procesos de SIAT	`adpsiat.jar`

Vista Física

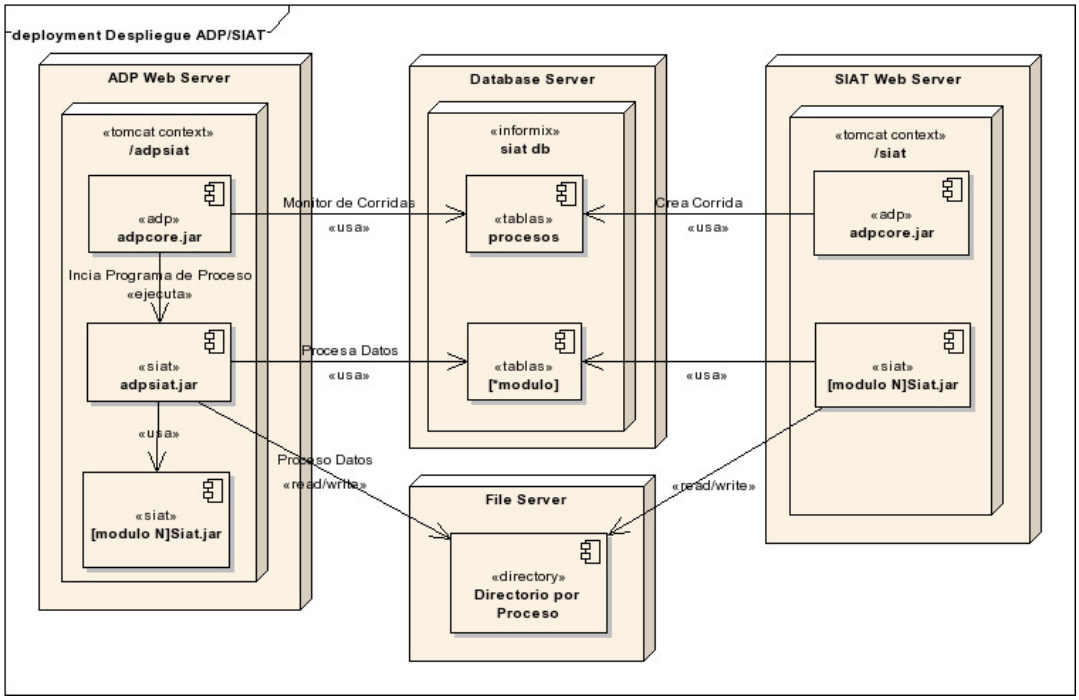
Estructura de despliegue

Ya que ADP es solo un marco que se utilizá para ejecutar programas que pertenecen a otra aplicación, cuando nos referimos a despliegue de ADP, en realidad nos referimos a una mezcla de componentes parte de adpcore y parte los procesos en si de SIAT.

ADP esta preparado para ser desplegado como parte de otra aplicación de tipo Web. Dicha aplicación debe contener las clases de procesamiento y configurar un servlet (AdpServlet) que es quien inicia ADP y prepara los Monitores. Para este fin creamos la aplicación **/adpsiat.**

Diagrama de despliegue

El siguiente es un diagrama de despliegue muestra ADP interactuando con SIAT.



El despliegue de ADP/SIAT cuenta con los siguientes elementos:

- ADP Web Server:
  Host donde se realiza la ejecución de los procesos de ADP de SIAT.
  Dicho Host posee el servicio Tomcat levantado, con la aplicación /adpsiat desplegada.
- El Contexto /adpsiat:
  Es la aplicación Web que realiza la ejecución de los procesos. Contiene el core de ADP y la lógica de los procesos SIAT.
- SIAT Web Server:
  El host que contiene la aplicación SIAT
- File Server:
  El host que hace de servidor de archivos para SIAT y ADP

El contexto de procesos SIAT "/adpsiat"

Para el caso SIAT, la aplicación en cuestión es la que llamamos "/adpsiat" que esta ubicada en directorio /adpsiat. Aquí dentro ademas de las clases de procesos del SIAT se hallan los dos archivos extras (web.xml y context.xml) que configuran el contexto /adpsiat

Ademas existen tareas ant para generar el WAR a desplegar. Básicamente el WAR debe contener:

- adpcore.jar:
  Las clases fundamentales de ADP, pej: AdpServlet, AdpProcess, etc.
- adpsiat.jar:

Las clases de procesos del SIAT

- [modulo N]Siat.jar
  Todos los jar de módulos de SIAT que requiera adpsiat.jar