B·Net.
JBC_Connect

24 de noviembre

2016

JBC Soldering SL

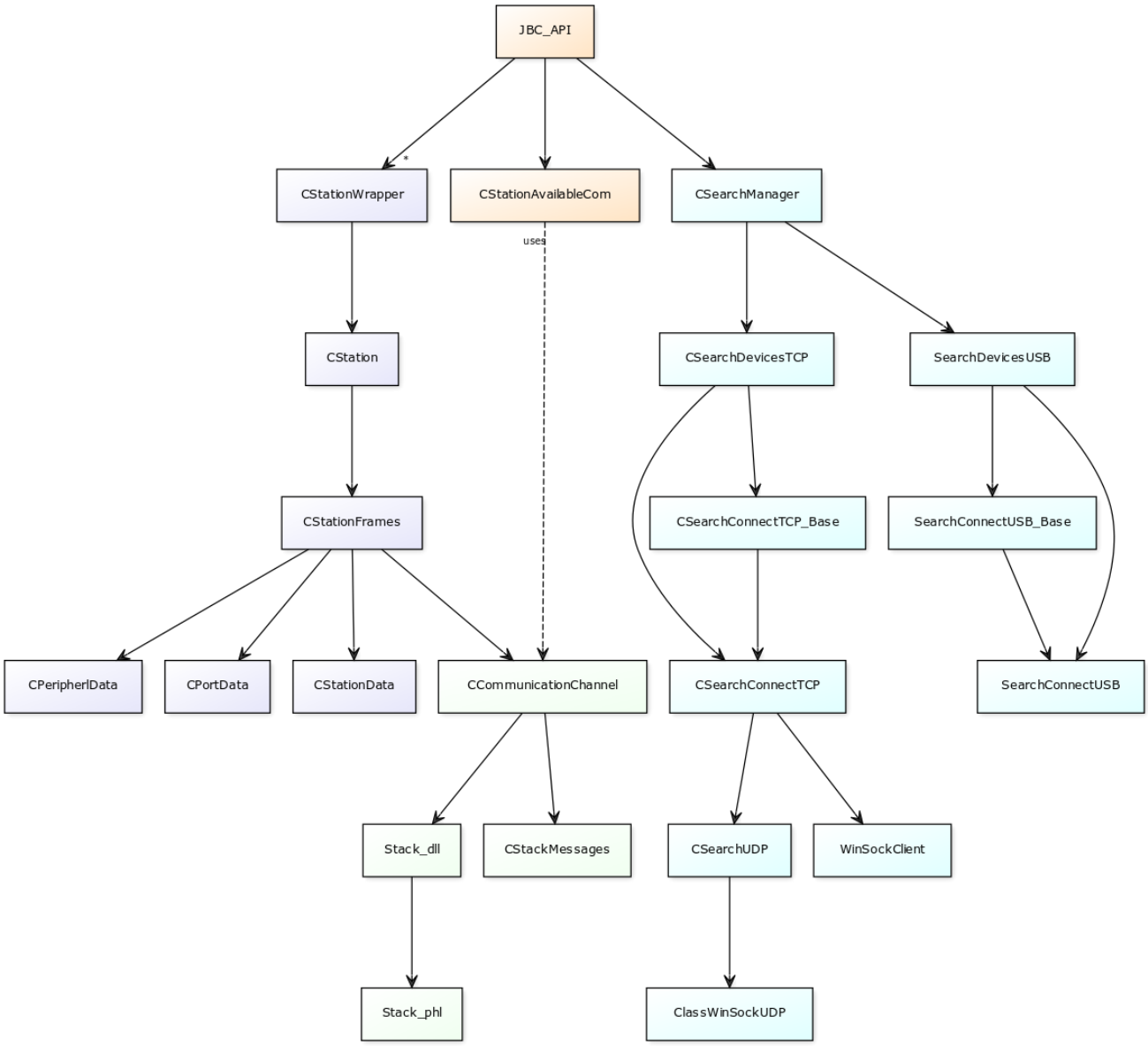
Histórico de versiones

Versión: **1.0.0**	*Autor:* **Andrés Di Giovanni**	*Fecha:* **24/11/2016**

Descripción: Versión inicial

Contenido

1 Arquitectura



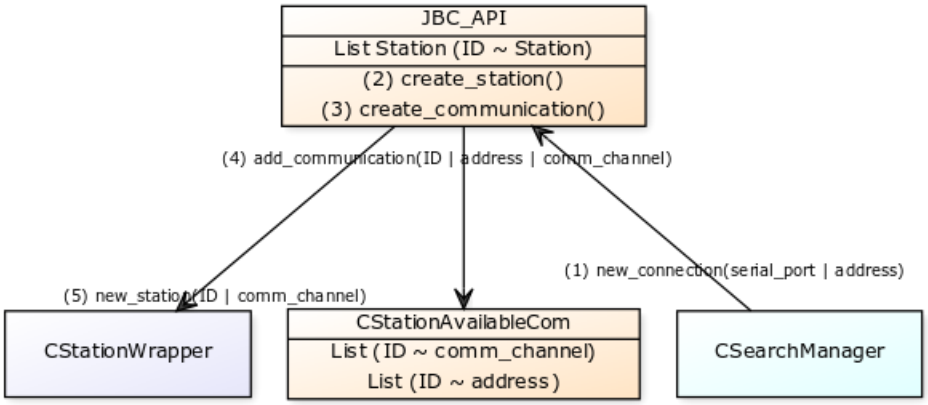
La aplicación se compone principalmente de 4 partes:

1. *JBC_API*. Es una interfaz que proporciona acceso a los métodos de la aplicación a través de una API. Contiene la lógica de control de las estaciones (creación y borrado), así como la relación entre la estación y el canal de comunicación que utiliza. Tiene acceso a la clase encargada de la búsqueda de nuevas estaciones.
2. *CStation*. Contiene toda la lógica de negocio referente a una estación. El punto de acceso es el *CStationWrapper* y permite abstraer el tipo de estación que se está utilizando (estación de soldadura, aire caliente, aspirahumos). Se compone principalmente de dos partes:
   2.1. *CStationsFrames*. Codifica y descodifica los frames según el protocolo utilizado. Tiene acceso a los datos de una estación y al canal de comunicación.
   2.2. *Data*. Almacenamiento de los datos de una estación, puertos y periféricos.

3. *CSearchManager*. Es la clase encargada de la búsqueda de nuevas estaciones.
4. *CCommunicationChannel*. Es la clase encargada de la comunicación con las diferentes estaciones conectadas por un canal de comunicación (serial port / winsock).

2 Conexiones

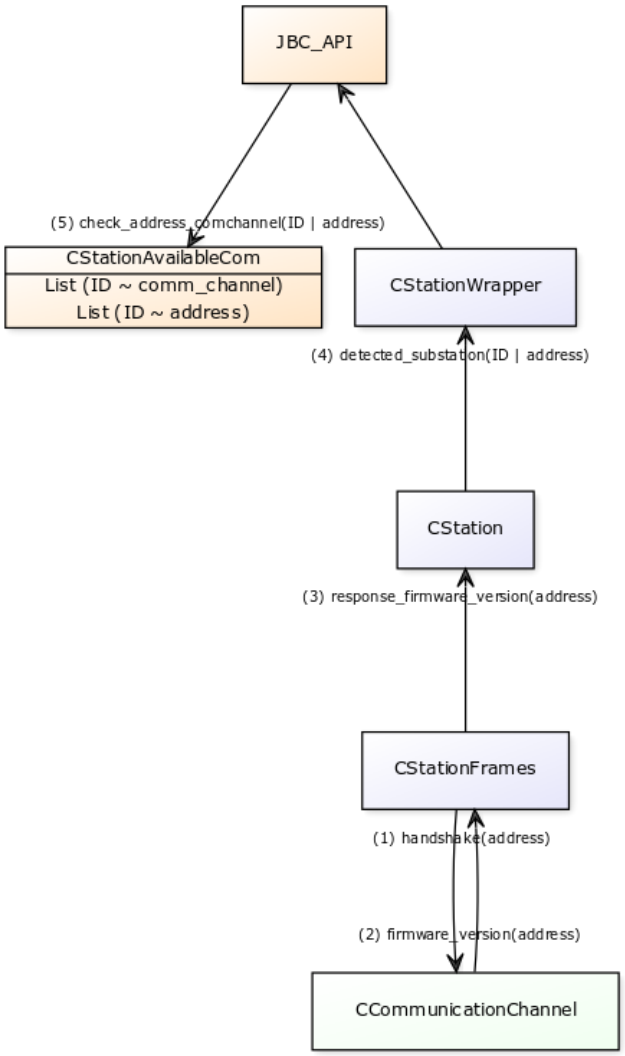
2.1 Nueva conexión



Proceso de nueva conexión:

1. El *CSearchManager* al detectar una nueva estación, notifica a *JBC_API*, pasándole, por referencia, el puerto de la nueva estación y la dirección del dispositivo (ej 0x10).
2. La *JBC_API* crea una nueva instancia de estación y le asigna un ID.
3. La *JBC_API* crea el canal de comunicación para la estación.
4. La *JBC_API* avisa al *CStationAvailableCom* de una nueva conexión enviándole el ID de la estación, el canal de comunicación y la dirección del dispositivo.
5. El *CStationWrapper* según el tipo de estación crea una instancia de la estación correspondiente (estación de soldadura, aire caliente, aspirahumos).

2.2 Nueva conexión sub-estación

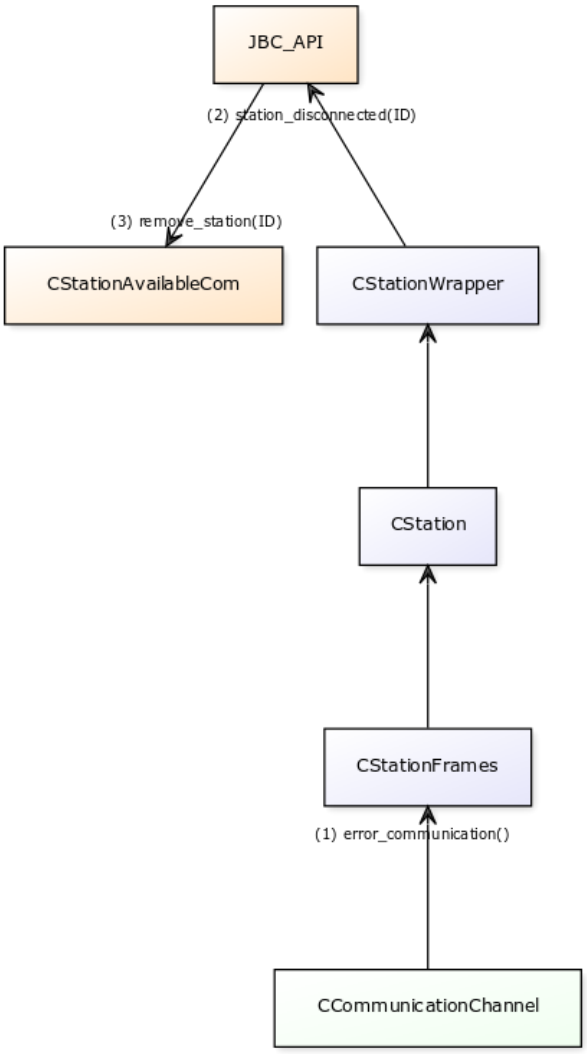


Proceso de nueva conexión de sub-estación:

1. Las estaciones conectadas, periódicamente, envían un *handshake* para notificar de su presencia. Como todas las estaciones conectadas en serie comparten canal de comunicación, todas reciben la trama de *handshake*, pero únicamente la siguiente estación de la cadena (determinado por el address) procesa la trama.
2. El *CStationFrames* al recibir un *handshake* envía una trama de leer el *firmware version*.
3. Cuando el *CStationFrames* recibe respuesta del *firmware version*, notifica con los datos del firmware.
4. El *CStation* notifica de una sub-estación, pasando su ID (para determinar el canal de comunicación) y el address de la sub-estación.
5. *JBC_API* al recibir la notificación de una nueva sub-estación, comprueba a través del *CStationAvailableCom* si la estación existe. Sino es el caso crea la nueva estación siguiendo el proceso habitual.

Nota: La conexión física de una estación (eth/usb) sólo se realiza con la primera estación (0x10).

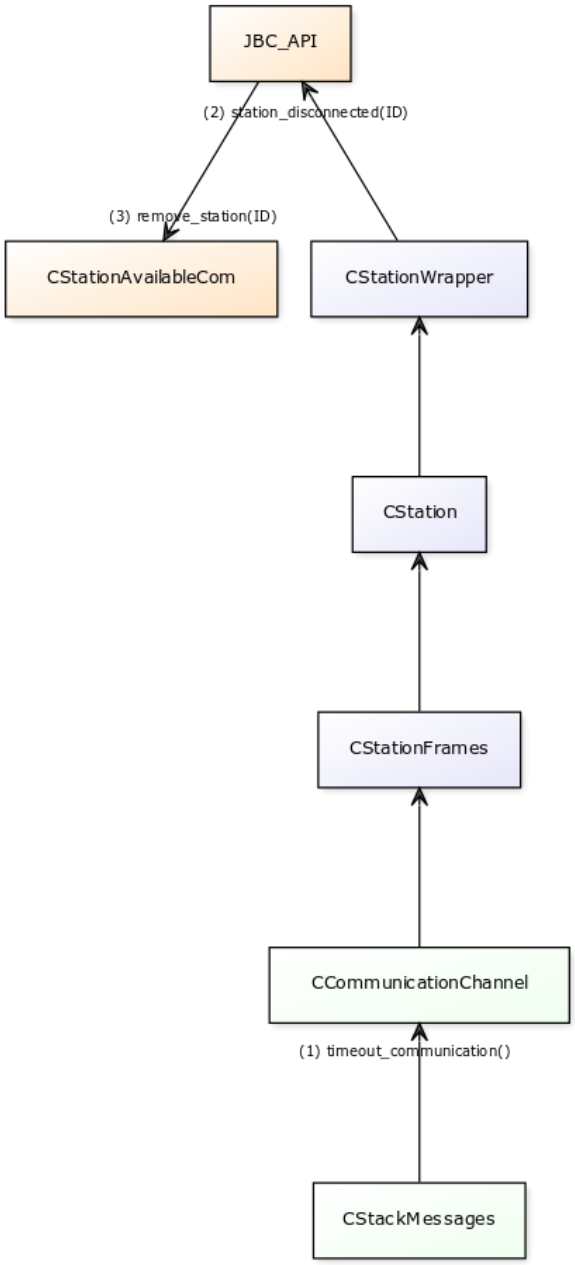
2.3 Desconexión



Proceso de desconexión:

1. Al desconectarse una estación físicamente, o al apagarse, se genera una excepción en la comunicación que es capturado con un try.
2. El error se propaga hasta notificar a la *JBC_*API de que una estación se ha desconectado. Liberando sus recursos y marcándola como desconectada.
3. La *JBC_API* comunica al *CStationAvailableCom* que elimine la estación. Y, éste, en caso de que no existan más estaciones en el mismo canal de comunicación, lo cierra.

2.4 Desconexión sub-estación



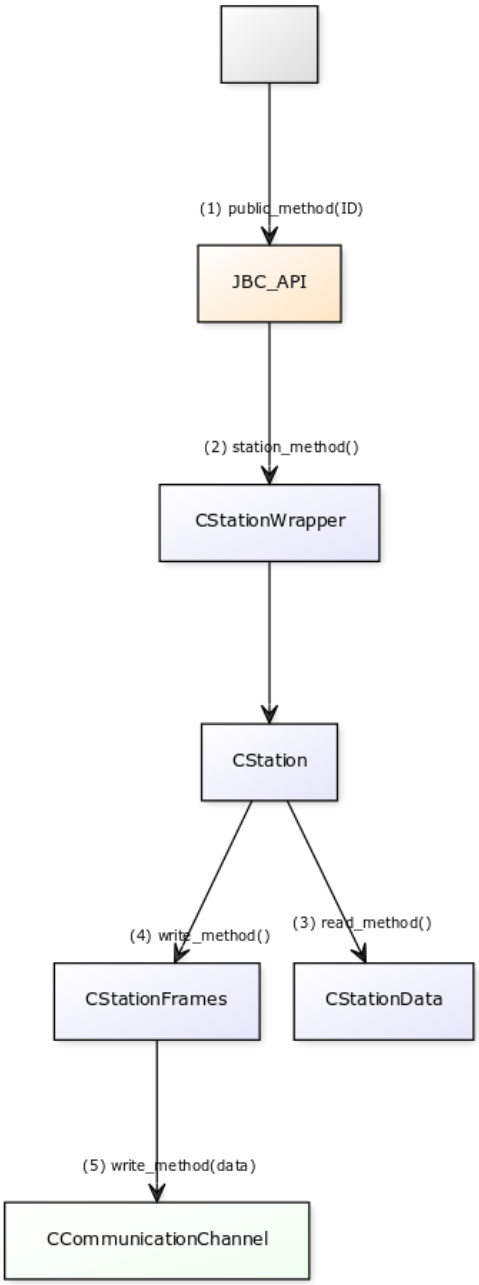
Proceso de desconexión de sub-estación:

1. Al desconectarse una sub-estación no se genera ninguna excepción en la comunicación, sin embargo, deja de existir un intercambio de datos. Pasado un tiempo desde la última comunicación, se genera un error que se propaga hasta *JBC_API*. Y, siguiendo el procedimiento habitual, se elimina la estación.

Nota: Si se envía una trama a una dirección no válida, la estación principal (0x10) no devuelve ningún mensaje. Por lo que es imposible detectar si una sub-estación se ha desconectado.

3 Comunicación

3.1 API comunicación



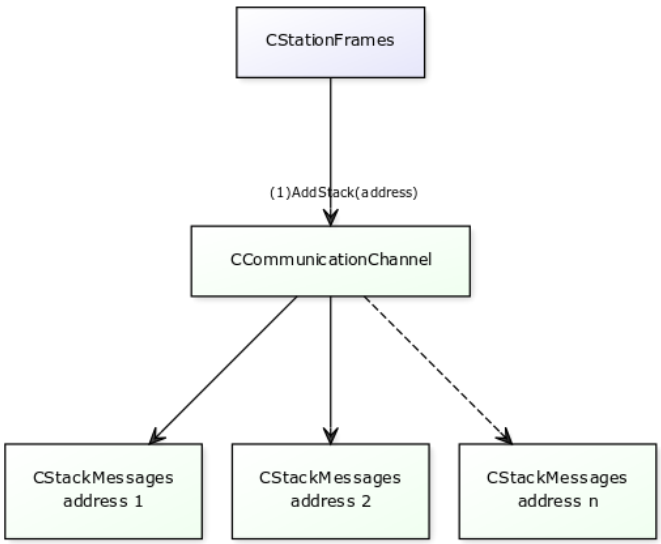
Proceso de comunicación con la API

1. La *JBC_API* recibe una llamada a un método público con el ID de la estación.
2. La *JBC_API* llama al método de la estación correspondiente.
3. Si es un método de lectura, responde con los datos recogidos del *CStationData*.
4. Si es un método de escritura, el *CStationFrames* procesa los datos a escribir y monta el frame según el protocolo que utilice la estación.
5. El *CStationFrames* envía al *CCommunicationChannel* los datos a enviar.

3.2 Gestión del canal de comunicación

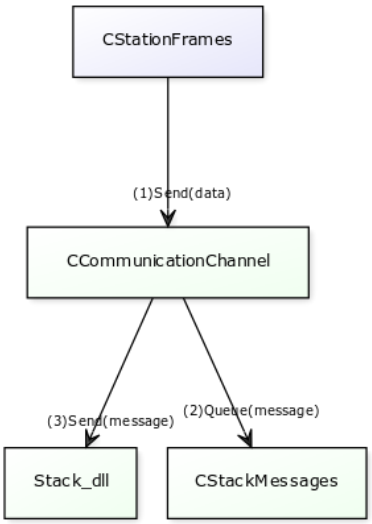
Un canal de comunicación (puerto usb/ethernet) puede ser compartido por múltiples estaciones (estaciones conectadas en cadena). Para cada estación, dentro de un mismo canal de comunicación, se crea una cola de mensajes en donde gestiona internamente qué mensaje enviar y las desconexiones.

Añadir cola de mensajes:



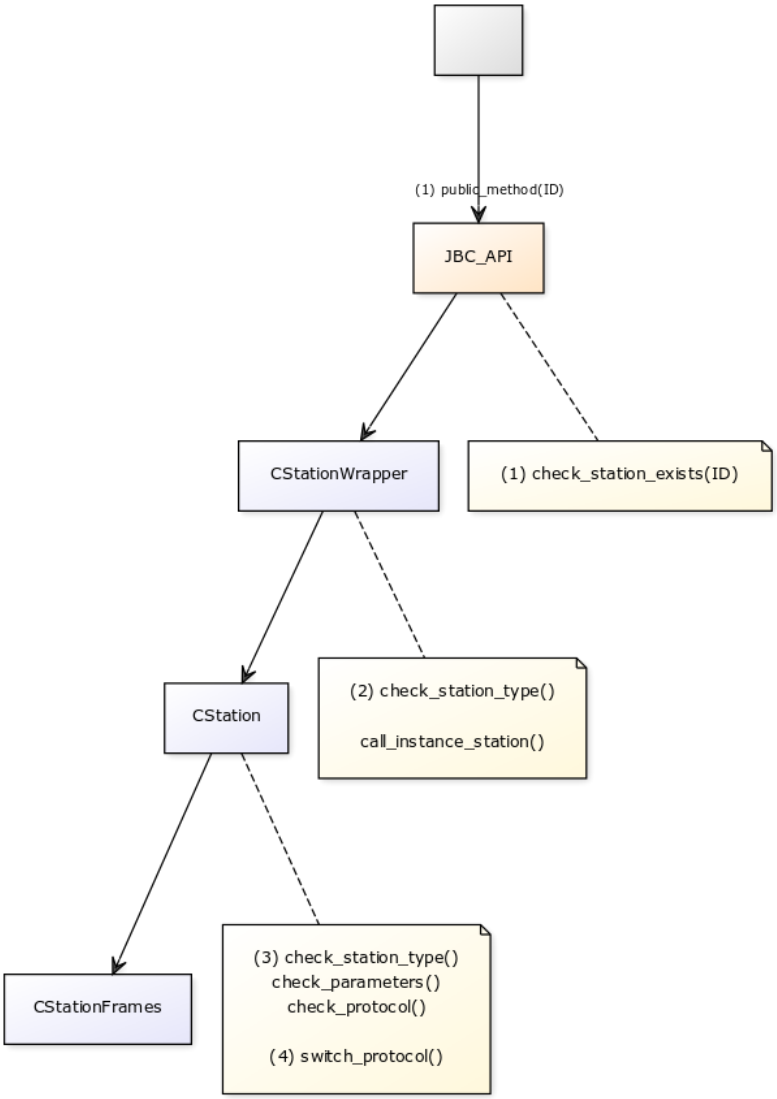
1. El *CStationFrames* envía una petición de añadir una cola de mensajes al *CCommunicationChannel* especificando la dirección que gestionará la cola.

Enviar mensaje:



1. El *CStationFrames* envía los datos del mensaje al *CCommunicationChannel* junto a la dirección destino.
2. Si la cola existe añade el mensaje a la cola correspondiente.
3. Si la cola no existe envía el mensaje directamente (también los mensajes de broadcast).

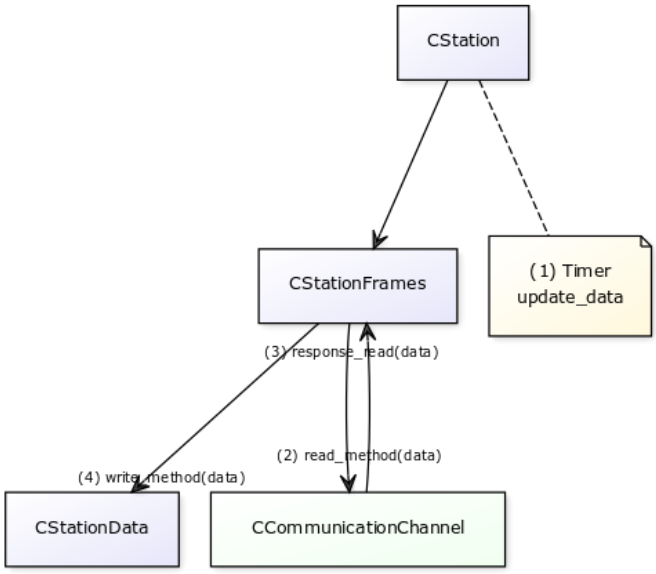
4 Soporte para los tipos de estación



1. La *JBC_API* comprueba si la estación existe.
2. *Station Wrapper* comprueba si el método existe para el tipo de estación y llama al método correspondiente para ese tipo de estación.
3. *Station* comprueba:
    a. Si la característica (feature) es soportada por el modelo de estación.
    b. Si los parámetros son válidos (n. puerto, …).
    c. Si, según el protocolo de la estación, el método es soportado.
4. El *Station* utiliza el *Station Frames* correspondiente según el protocolo de la estación.

Nota: Cada tipo de estación tiene unas características distintas (features).

5 Actualización de los datos de la estación



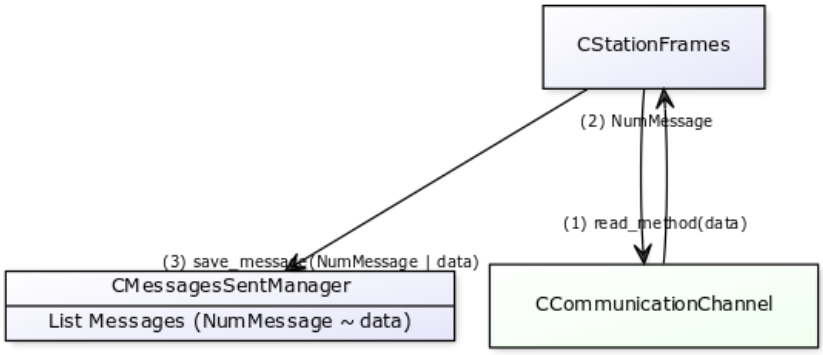
Proceso de actualización de los datos de la estación

1. *CStation* dispone de un timer que periódicamente utiliza para actualizar los datos de la estación, con una frecuencia variable según los datos que se quieran actualizar.
2. *CStation* llama a *CStationFrames* para que monte el frame según el protocolo que utilice la estación. Y este, envía al *CCommunicationChannel* el frame.
3. El *CCommunicationChannel* responde con los datos de la estación.
4. El *CStationFrames* procesa los datos y almacena el *CStationData* los datos.

6 Recuperación de datos enviados

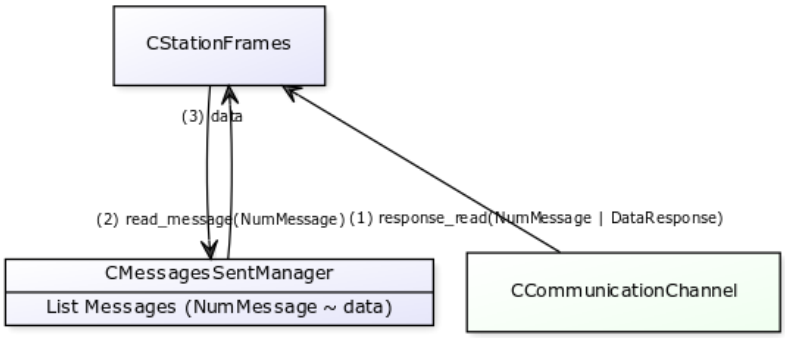
En el protocolo 1 de las estaciones, cuando se reciben los datos de una petición de lectura, no viene indicada en la respuesta a que puerto y herramienta se hizo la consulta. Para solventar el problema se ha ideado el siguiente mecanismo:

Guardar mensaje enviado:



1. El *Station Frames* envía los datos que quiere enviar al *Communication Channel*.
2. El *Communication Channel* devuelve un número de mensaje para esa petición.
3. El *Station Frames* guarda el número de mensaje con los datos del mismo.

Recuperar mensaje enviado:



1. Al recibir los datos de respuesta de una petición de lectura, el *Communication Channel* responde además con el número de mensaje de esa trama.
2. El *Station Frames* recupera, de los mensajes que envió, el correspondinte a ese número de mensaje.
3. El *Station Frames* obtiene los datos de petición (puerto y herramienta).

Nota: El Messages Sent elimina periódicamente los mensajes de la lista que llevan mucho tiempo sin respuesta.

7 Transaction ID

SetTransaction:

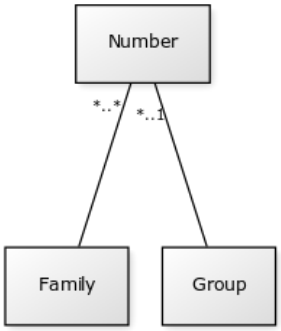
Cuando se recibe una petición de *SetTransaction*, *CStation* envía un mensaje de *ACK* a la estación y responde a la petición con el número de mensaje que ha enviado.
Al recibir la respuesta del *ACK* procedente de la estación, el *CStationFrames* guarda el número de mensaje en una lista de transacciones finalizadas.

QueryTransaction:

Cuando se recibe una consulta a una transacción finalizada, *CStation* consulta si el número de mensaje lo tiene guardado como transacción finalizada.

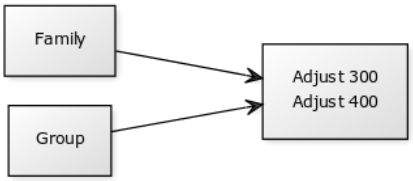
8 Cartridges

Los cartuchos se identifican por un *número* y por la *familia* a la que pertenecen. Y a su vez, cada cartucho pertenece a un *grupo*.



Ejemplo: *C210-001*. Cartucho 001 de la familia C210

Según la relación de *grupo-familia* que pertenece un cartucho se le aplica una corrección de temperatura (300ºC – 400ºC).

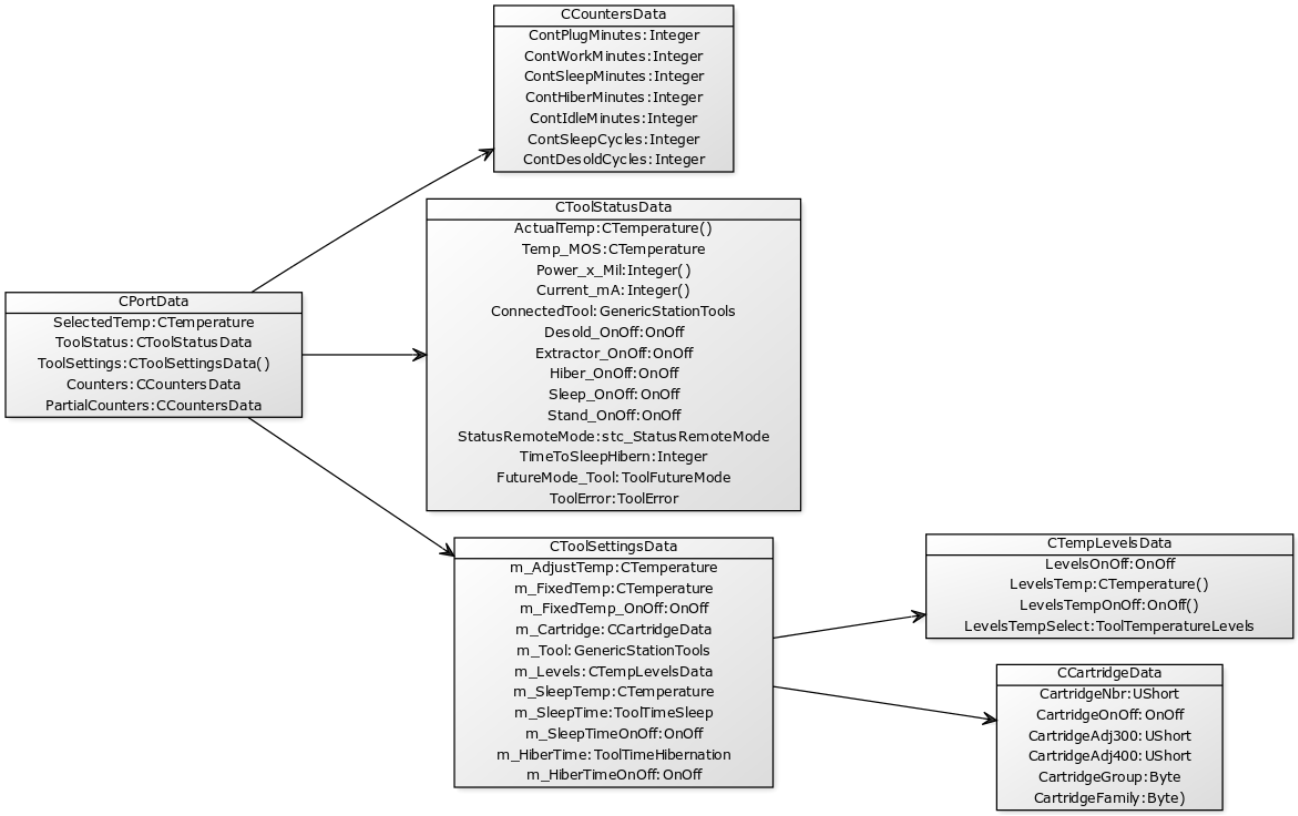


La familia a la que pertenece un cartucho viene dada según la herramienta y a la estación a la que esté conectada.

Familia	Herramienta	Estación
C105		
C120	PA	
C130	AP	
C210	T210	~HD, ~HDR
C245	T245	
C250	AP	
C360	DS	
C420	HT	
C470	T210	HD, HDR
C560	DR	

9 Data

9.1 Port Data



9.2 Station Data