**Allgemeines**

Das DeviceFramework (kurz DF) ist für das Handling der verschiedenen Leser zuständig, welche im TIM-Frontend und TIM-Administrator verwendet werden. Hierfür wird ein Windows Communication Foundation Service (kurz WCF) genutzt. Dieser läuft lokal unter der Adresse [***http://localhost:54321/DeviceFramework***](http://localhost:54321/DeviceFramework).

Zum Start von TIM’s Devices muss das Projekt ***TIM-Devices*** als Start-Projekt gewählt werden.

**IDevice.cs**

Jedes Gerät erbt von ***IDevice***. IDevice gibt zwei Events vor, MediaDetected und MediaRemoved. Die Events werden von dem lokalen Server abonniert und an alle Clients geschickt. MediaDetected wird aufgerufen, sobald eine Karte erkannt wurde, und wird von jedem Gerät unterstützt. MediaRemoved hingegen wird genutzt, wenn eine Karte vom Leser entfernt wurde. Dieses Event wird ausschließlich von den Geräte ***Feig Reader und GAT Writer*** unterstützt.

***Name:*** Hier wird der Name angegeben, welcher in TIM’s Devices angezeigt wird.

***MediaSearchEnabled:*** Ist MediaSearchEnabled auf True, sucht das Gerät nach Karten oder Schlüssel.

***ConfigString:*** Wenn das Frontend für den Self-check-in Geräte ansteuert, wird hier die benötigte Konfiguration gespeichert.

***DeviceIsOpen:*** Gibt an, ob ein Gerät erfolgreich geöffnet wurde oder nicht.

***DeviceID:*** Die DeviceID gibt an, um welche Art Gerät oder Chip Technologie es sich handelt. Diese ID wird auch für ***tblCheckInIDs*** oder ***CustomerKeys*** benötigt. Für alle Geräte von Gantner wird nur zwischen ***Legic*** und ***Mifare*** unterschieden.

***Message:*** Message ist die Nachricht, die auf dem Display des Geräts angezeigt wird. Aktuell nur unterstützt vom GT7.

***IsConnected:*** Gibt bei Netzwerkgeräten an, ob diese aktuell verbunden sind. Bei anderen Geräten wird das Property ***DeviceIsOpen*** zurückgegeben.

***DeviceFrameworkDeviceID:*** Diese DeviceID ist zur internen eindeutigen Zuordnung der Geräte.

***SupportsUnblockLockerKey:*** Wenn True, kann mit diesem Gerät das Freigeben des Chips für die Schränke veranlasst werden.

**TIM-Devices**

Im Konstruktor werden die ***ConfigDevicesInHouse***, ***ConfigDevicesExternal*** und ***ConfigDevicesEmployee*** gefiltert nach Anlage und Aktivität geladen. Für jedes Gerät muss geprüft werden, ob es in einer der Listen vertreten ist. Andernfalls darf das Gerät nicht von TIM’s Devices geladen werden.

**FeigIDRWA02.IDRWA02.cs**

Der Feig Leser wird aktuell nur noch in Kiel für das Drehkreuz in Kombination mit dem Self Check-in genutzt.

**Trigger(TimeSpan tsActive):** Aktiviert das Drehkreuz. “tsActive” gibt an, wie lange das Relais getriggert werden soll.

**GantnerAccess6200.Access6200.cs**

**MyGATAccess\_CardIdentificationRequested():** Wird aufgerufen, sobald ein Schlüssel/Chip an den Leser gehalten wird. Zurückgegeben wird ein “CardIdentificationGATAccessResult”, wo das Feld “IsAuthorized” angiebt, ob der Benutzer erfolgreich authentifiziert werden konnte. Im Feld “Texts” kann ein Text übergeben werden, der –falls vorhanden- auf dem Display des Geräts angezeigt wird.

**FunWriter.cs**

Die GATWriter sind die am meisten genutzten Geräte. Früher wurden diese noch „FunWriter“ genannt. Inzwischen hat sich der Name geändert und es sollte auch nur noch mit dem neuen Namen gearbeitet werden. Diese werden zum manuellen Check-in, TIM-Login, Mitglieds- oder Mitarbeiterkarten einer Person zuweisen oder im Restaurant verwendet. Hier sind drei verschiedene Modelle im Einsatz:

GATWriter5200: Das älteste Modell und liest Legic

GATWriter5250: Das aktuelle Modell und liest Legic

GATWriter6000: Das aktuelle Modell und liest Mifare

**LoadConfig():**

Hier werden die Properties „FID“, „SubFID“ und „Key“ gesetzt. Diese werden bei der Initialisierung der GATWriter verwendet und werden benötigt, um mit den Geräten Legic- und Mifare-Chips beschreiben zu können. Diese Funktion wird aktuell zum Setzen des Timestamps genutzt.

**Open():**

Bei der erstmaligen Initialisierung werden die LEDs der GATWriter gesetzt. Das grüne Licht wird eingeschaltet und das rote ausgeschaltet.

**Close():**

Beim Schließen des Gerätes werden alle Lichter ausgeschaltet.

**GATWriterUnblockLockerKey():**

Beschreibt den aufgelegten Chip mit einem Timestamp. Somit wird erlaubt oder verhindert, dass offline Schrankschlösser genutzt werden können. **WICHTIG**: Wenn der Timestamp auf den Default-Wert gesetzt werden soll, immer nur das „ExpiryDate“ auf einen Wert in der Vergangenheit setzen (z.B. zwei Jahre). Die GT7 können nur das Datum setzen und nicht die Config auf dem Chip neu schreiben, womit Nutzbarkeit des Chips auch gesteuert werden kann.

**GetLockerInfo()**

Liest die aktuell verschlossenen Schränke vom Chip und gibt diese als Json zurück.

**FunWriter5200.cs**

**MyTimer\_Elapsed():**

Diese Methode wird regelmäßig von einem Timer aufgerufen. Hier wird versucht, von dem aktuell aufgelegten Chip die ID zu laden.

**FunWriter5250.cs/FunWriter6000.cs**

Diese Leser nutzen keinen Timer um die Chip ID zu laden. Hier wird auf die DLLs von Gantner zurückgegriffen, die die Events „OnCardDetected“ und „OnCardLost“ bereitstellen. Für den GATWriter5200 ist diese Funktion nicht verfügbar.

**KeyboardListener.cs**

Der KeyboardListener wird für Magnetkartenlesegeräte gebraucht. Auf den Magnetkarten (MK) ist eine eindeutige ID gespeichert. Das Format (Regex) der ID kann in den allgemeinen Einstellungen von TIM festgelegt werden. Sobald der MK-Leser neue Kartendaten ausgelesen hat, werden diese als Tastenanschläge von Windows gewertet. Der KeyboardListener hört also auf die Tastatureingabe und prüft, ob die Eingabe zu schnell für Menschen war („KeyStroke()“). Anschließend werden die Daten auf das Format geprüft (CheckMediaDetected()) und ggf. an „OnMediaDetected“ weitergegeben.

**DS6707.cs/Motorola**

Motorola ist der genutzte Barcodescanner. Dieser wird überwiegend im PoS und Rezeption verwendet. Im PoS werden die Barcodes von Artikeln eingescannt und für eine Person gebucht. Beim Check-in wird die Datamatrix der Mitgliedskarten gescannt, da diese keinen Chip enthalten. Hier enthält die Datamatrix die Mitgliedsnummer.

**MyScanner\_Decode():**

Wird aufgerufen, sobald ein Code eingescannt wurde. Anschließend wird geprüft, um welche Art Code es sich handelt. Die Daten werden ausgelesen und das Event „OnMediaDetected“ wird aufgerufen.

**DeviceServer.cs**

Der DeviceServer wurde mit Windows Communication Foundation (WCF) umgesetzt. Der Service muss über alle vorhandenen Methoden, die auf Server- und Clientseite informiert sein. Dafür wird das Interface „IDeviceServer“ genutzt. Hier werden alle Methoden definiert und mit dem Attribut „OperationContract“ versehen. Um den Service zu aktualisieren, muss TIM’s Devices im Dev-Modus gestartet werden.

**Wichtig:** ohne Debugger, sonst können keine Änderungen an den Dateien gemacht werden. Um den Service zu aktualisieren, gehen Sie im Solution Explorer mit einem Rechtsklick auf „DeviceFramework.ServiceReferences.TIMsDevices“, anschließend auf „Update Service Reference“. Der Service nimmt nun alle Änderungen automatisch vor und speichert diese. Nun ist nur noch ein Rebuild der genutzten Solutions nötig und die neuen oder geänderten Methoden können genutzt werden.

**MyDeviceManager\_MediaDetected():**

Hier kommen die Events an, die von den Geräten ausgelöst werden. Es gibt eine neuere und eine ältere Version von dem Event „FireMediaDetected“ welches dann an alle Clients geschickt wird. Es ist nicht klar, ob das ältere Event noch genutzt wird. Um Komplikationen zu vermeiden, wurde der Code bisher nicht gelöscht.

**MyDeviceManager\_MediaRemoved():**

Gleiches Vorgehen wie bei MediaDetected.

**RemoveClients():**

Die Verbindung zum Client wurde getrennt und der Client wird nun aus der Liste entfernt.

**AddIDeviceClient():**

Neuer Client hat sich beim Server angemeldet.

**GetDeviceNames():**

Gibt ein Array mit den Namen aller initialisierten Geräte zurück.

**DeviceClient.cs**

Der DeviceClient wird in der TIM-Solution initialisiert. Dieser verbindet sich mit dem Server und hört dann auf die Nachrichten des Servers.

**Connect():**

Verbindet sich mit dem Server.