|  |  |
| --- | --- |
|  | **ETCS@EBD** |
|  | RBC – TMS Kommunikation  Schnittstellendefinition |
|  |  |
|  |  |
|  | Konzeptpapier, v0.3  Status: DRAFT, Stand: 04.05.2020 18:42  Autor: Christopher Bernjus |
|  |  |



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Bearbeitung Editorial** | **Qualitätsprüfung Reviewer** | **Freigabe Approval** |
| **Name** | Christopher Bernjus | Werner Iberl | Dr. Georg F. Bolz |
| **Position/Rolle** | Mitarbeiter (HiWi) | Mitarbeiter (HiWi) | Teamleiter ETCS@EBD |
| **Datum** | 18.03.2020 |  |  |
| **Unterschrift** | gez. Bernjus |  |  |

Versionshistorie

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vers.** | **Datum** | **Kap.** | **Beschreibung** | **Autor** |
| 0.1 | 03/18/20 |  | Erste Draft Version | CB |
| 0.2 | 03/21/20 | 1  2, 3, 4 | Einleitung erweitert  Zeitbehandlung angepasst  Formulierungen konkretisiert | CB |
| 0.3 | 05/04/20 | 6 | Ablaufdiagramme hinzugefügt  Inhaltliche Überarbeitung | CB |

1 Einleitung 7

2 Format und Struktur 8

2.1 Header 8

2.1.1 Type 8

2.1.2 UUID 8

2.1.3 Hash 8

2.1.4 IDs 9

2.1.5 Zeitangaben 9

2.1.6 Protokollversion 9

2.2 Payload 9

2.2.1 TrainInfo 9

2.2.2 TrainData 10

2.2.3 PositionInfo 10

3 Allgemeine Nachrichten 11

3.1 Error 11

3.2 Register 11

3.3 Unregister 12

4 RBC Nachrichten 13

4.1 LogIn 13

4.2 LogOut 13

4.3 Time Factor Changed 13

4.4 TrainData 14

4.5 Position Report 14

4.6 MA Request 14

4.7 SH Request 15

5 TMS Nachrichten 16

5.1 Position Report Request 16

5.2 MA 16

5.3 Request to Shorten MA 18

5.4 Conditional Emergency Stop 19

5.5 Unconditional Emergency Stop 20

5.6 Revocation of Emergency Stop 20

6 Kommunikationsabläufe 21

6.1 Register und Unregister 21

6.2 Train Not Managed 22

6.3 Position Report Request 23

6.4 Probleme beim Position Report 24

6.5 MA Request 25

Verzeichnisse 26

Listingverzeichnis 26

Literaturverzeichnis 27

Glossar 28

Anlagen 29

Anlage 1: Logos des ETCS Projekts 29

# Einleitung

Das Traffic Management System (ab hier kurz TMS) und das ETCS Radio Block Center (ab hier kurz RBC) sollen miteinander kommunizieren können. Das TMS soll in der Lage sein korrekte Movement Authorities für Züge auszustellen, jedoch mit so wenig Informationen wie möglich über die interne ETCS Kommunikation zwischen Zug und RBC.

Die benötigten Daten richten sich dennoch nach dem Format der ETCS Spezifikation (in der Version 3.4.0), wie es in [SRS 7] und [SRS 8] definiert ist.

Um eine reibungslose Kommunikation zu gewährleisten wird der Datenaustausch in einem standardisierten Datenformat (JSON) abgewickelt. Dabei wird in einer Übertragung genau eine Nachricht als zu einem String konvertierten JSON Objekt versendet. Das TCP Protokoll sorgt für die korrekte Zustellung.

Das Protokoll geht davon aus, dass beide Seiten vor Kommunikationsbeginn die jeweilige IP und den Port für die Kommunikation kennen.

Diese Protokollversion unterstützt zurzeit nur die Kommunikation zwischen einem TMS und einem oder mehreren RBCs. Die Kommunikation zu mehreren TMSs muss zu einem späteren Zeitpunkt genauer untersucht und spezifiziert werden.

# Format und Struktur

Dieser Abschnitt definiert JSON Objekte, die in den einzelnen Nachrichten verwendet werden. Werte können vom Typ string, number (hier genauer unterteilt in int und long), Object (mit geschweiften Klammern gekennzeichnet) oder Array (mit eckigen Klammern gekennzeichnet) sein. Manche Werte sind optional und können daher auch den Wert null annehmen.

## Header

Der Header enthält die zur Kommunikation nötigen Informationen und wird in jeder Nachricht mitgeschickt.

### Type

Der Nachrichtentyp („type“) dient zur Kennzeichnung der Nachrichtenart und bestimmt den zu erwartenden Inhalt der Payload.

Der Nachrichtentyp wird als integer angegeben und kann einen der folgenden Werte annehmen:

type := 00 // Error

| 01 // Register

| 02 // Unregister

| 10 // Log In

| 11 // Log Out

| 12 // Time Change

| 13 // Train Data

| 14 // Position Report

| 15 // MA Request

| 16 // SH Request

| 20 // Position Report Request

| 21 // MA

| 22 // Request to Shorten MA

| 23 // Conditional Emergency Stop

| 24 // Unconditional Emergency Stop

| 25 // Revocation of Emergency Stop

Listing 1: Message Type Definition

### UUID

Die UUID (**„uuid“**) (Universally Unique Identifier) dient der Kennzeichnung und Nachverfolgung von Kommunikationsabläufen. Eine Antwort verwendet somit immer die UUID der ursprünglichen Nachricht oder Anfrage. Eine Konversation umfasst alle zu einer UUID versendeten Nachrichten. Für eine Nachverfolgung müssen die versendeten und empfangenen Nachrichten einer Konversation gespeichert werden. Sobald eine Konversation beendet wurde können die gespeicherten Nachrichten verworfen werden und die UUID darf wiederverwendet werden.

Die UUID wird als string angegeben.

### Hash

Der Hash (**„hash“**) wird bei der Erzeugung einer Nachricht über der Payload generiert und soll eine korrekte Übertragung sicherstellen. Dieser Wert kann später auch für die Simulation von Übertragungsfehlern hilfreich sein.

Der Hash wird als integer angegeben.

### IDs

Eine Kommunikation findet immer zwischen einem TMS und einem RBC statt. Das TMS wird über die **„tms\_id“** und das RBC über den Wert **„rbc\_id“** adressiert. In beiden Variablen kann der Flag „all“ verwendet werden, falls einer der Kommunikationspartner noch nicht bekannt ist (beispielsweise bei der Register Nachricht)

Beide werden als string angegeben.

### Zeitangaben

Der Header einer Nachricht enthält eine Zeitangabe für die Nachrichtenerstellung („**timestamp**“). Diese bezieht sich auf die Realzeit und gibt daher Zeitpunkte nach der Java Definition in Relation zum 01.01.1970 UTC an.

Zeiten werden als long angegeben.

### Protokollversion

Der Header enthält außerdem einen Wert („**version**“), der die letzte unterstützte Version des hier definierten Protokolls angibt.

header := {

"type": type,

"uuid": string,

"hash": int,

"tms\_id": string | "all",

"rbc\_id": string | "all",

"timestamp": long

}

Listing 2: Header Definition

## Payload

Die Payload („payload“) ist ein JSON Objekt und umfasst alle spezifischen Informationen, die in einer Nachricht benötigt werden. An dieser Stelle werden zwei in RBC Nachrichten wiederkehrende Objektstrukturen definiert.

### TrainInfo

TrainInfo („trainInfo“) enthält die Informationen, die zur Identifizierung eines bestimmten Zuges notwendig sind.

**“nid\_engine”** identifiziert im gesamten ETCS System eindeutig ein Fahrzeug.

**„nid\_operational“** wird vom Zugführer vergeben und ist die Running Train Number des jeweiligen Zuges. Diese wird auch vom Stellwerk und dem Fahrplantool verwendet.

**„t\_train“** gibt den Zeitpunkt in AppTime an. Dieser bezieht sich auf die zuvor übergebene „**time\_base**“ in einer Register oder TimeChanged Nachricht.

trainInfo := {

"nid\_engine": int,

"nid\_operational": int,

"t\_train": long // in AppTime

}

Listing 3: TrainInfo Definition

### TrainData

Das TrainData Object wird benutzt, um die im Zug hinterlegten Validated Train Data zu übertragen.

"trainData":= {

"nc\_cdtrain": number,

"nc\_train": number,

"l\_train": number,

"v\_maxtrain": number,

"m\_loadinggauge": number,

"m\_axleloadcat": number,

"m\_airtight": number,

"n\_axle": number,

"tractionSystems": [ {

"m\_voltage": number,

"nid\_ctraction": number | null

} ],

"nationalSystems": [ {

"nid\_ntc": number

} ]

}

### PositionInfo

Das PositionInfo Object umfasst alle Informationen, die zur Positionsbestimmung eines Zuges notwendig sind.

Die Bedeutung und Wertebereiche der jeweiligen Werte sind identisch zur ETCS Spezifikation [SRS 7].

positionInfo := {

"q\_scale": int,

"nid\_lrbg": int,

"nid\_prvlrbg": int | null,

"d\_lrbg": int,

"q\_dirlrbg": int,

"q\_dlrbg": int,

"l\_doubtover": int,

"l\_doubtunder": int,

"q\_length": int,

"l\_trainint": int | null,

"v\_train": int,

"q\_dirtrain": int,

"m\_mode": int,

"m\_level": int,

"nid\_ntc": int | null

}

Listing 4: PositionInfo Definition

# Allgemeine Nachrichten

## Error

Die Error Nachrichten übertragen einen ErrorCode der als integer angegeben wird und einen der folgenden Werte annehmen kann:

errorCode := 00 // Accepted | No Error

| 01 // Rejected

| 02 // Invalid Message

| 03 // Unmanaged Train Number

| 04 // Unknown

| 10 // Connection to Train Interrupted

Listing 5: Error Type Definition

Eine Error Nachricht mit dem Error Code 00 wird verwendet, um eine Nachricht oder Anfrage zu bestätigen.

Eine Error Nachricht mit dem Error Code 01 wird benutzt, um eine Nachricht oder Anfrage abzulehnen.

Wenn eine empfangene Nachricht fehlerhafte Daten enthält wird als Antwort eine Error Nachricht mit dem Error Code 02 versendet. Diese identifiziert die fehlerhafte Nachricht über die UUID im Header.

Wenn eine Kommunikation bzgl. einer Fahrzeugnummer („nid\_engine“) stattfindet, die von einem der Kommunikationspartner nicht verwaltet wird, versendet dieser eine Error Nachricht mit dem Error Code 03.

Falls ein allgemeiner Fehler in einem der kommunizierenden Module auftritt, die die Kommunikation zwischen dem TMS und dem RBC beeinträchtigen, muss die Gegenseite darüber informiert werden. Dazu wird eine Error Nachricht mit dem Error Code 04 versendet.

Falls die Kommunikation zwischen dem RBC und einem Zug zusammenbricht (d.h. der Zug auf Nachrichten, die eine Antwort der ETCS On Board Unit oder des Triebfahrzeugführer erfordern, nicht rechtzeitig antwortet), muss das TMS mithilfe einer Error Nachricht mit dem Error Code 10 informiert werden.

{

"header": header,

"payload": {

"error\_code": errorCode,

“trainInfo”: trainInfo | null

}

}

Listing 6: Error JSON Format

## Register

Die Register Nachricht kann von jedem Kommunikationspartner verwendet werden, um sich für eine Kommunikation bereitzustellen.

Dafür sendet z.B. das RBC eine Register Nachricht mit dem All-Flag (angegeben als string: „all“) als „**tms\_id**“ (analog würde das TMS „**rbc\_id**“ verwenden) um sich an allen TMSs (oder RBCs) zu registrieren. Auf diese Weise erfährt es auch über alle möglichen Kommunikationspartner. Falls schon ein Kommunikationskanal zwischen den Kommunikationspartnern etabliert wurde, wird der Error Code 00 erneut gesendet.

Damit die ETCS-Applikation mit gegenüber der Realzeit beschleunigter oder verzögerter Zeit ablaufen oder zwischenzeitlich anhalten kann, und da nur die RBC im Kontext der Applikation laufen und die Programmzeit (AppTime) von der zentralen AppTimeClock übernehmen können, muss ein RBC das TMS über die AppTime informieren.

Dazu übermittelt es einen Bezugszeitpunkt (**„time\_base“**), ab dem die AppTime verändert läuft, und einen Umrechnungsfaktor (**„time\_factor“**), um den die Realzeit skaliert werden muss, um die Programmzeit zu erhalten, die zur Angabe von Zeiten in TrainInfo verwendet wird. Somit ergibt sich:

time\_app = time\_base + (time\_real – time\_base) ⋅ time\_factor.

Die von einem RBC versandte Register Nachricht übermittelt die aktuell gültigen Werte für **„time\_base“** und „**time\_factor**“ in der Payload. Die von einem TMS versandte Register Nachricht übermittelt **„time\_base“** = null und „**time\_factor**“ = 1.0.

Die beiden Zeitangaben **„time\_real“** und **„time\_base“** geben die seit Mitternacht am 1. Januar 1970 UTC (unix epoch) abgelaufene Zeit in Millisekunden an.

Der „**time\_factor**“ kann größer (Beschleunigung) oder kleiner (Verlangsamung) als 1.0 sein. Er muss positiv sein.

{

"header": header,

"payload": {

"time\_base": long | null,

"time\_factor": int,

"version": string

}

}

Listing 7: Register JSON Format

## Unregister

Die Unregister Nachricht dient zum Abmelden im Kommunikationskanal zwischen TMSs und RBCs.

Der Ablauf für eine Abmeldung an allen Kommunikationspartnern funktioniert analog zur [Register](#_Register) Nachricht. Jedoch kann eine Instanz sich auch bei einem konkreten Kommunikationspartner unregistrieren. Dafür wird die Unregister Nachricht an eine bestimmte „tms\_id“ bzw. „rbc\_id“ gesendet.

Falls noch Züge angemeldet sind kann das RBC sich beim TMS nicht unregistrieren. Das TMS kann dagegen den Abbruch der Verbindung zum RBC mit der Unregister Nachricht anfragen.

{

"header": header,

"payload": {}

}

Listing 8: Unregister JSON Format

# RBC Nachrichten

## LogIn

Sobald ein mit ETCS aufgerüsteter Zug mit einem RBC Verbindung aufnimmt und diese vom RBC akzeptiert würde, schickt das RBC eine LogIn Nachricht an das TMS, um über die Aufnahme des neuen Zuges zu informieren.

Der Empfang dieser Nachricht wird vom TMS mit einer Error Nachricht quittiert. Akzeptiert das TMS die LogIn Nachricht mit dem Error Code 00, so kann die Verbindung zwischen Zug und RBC bestätigt und etabliert werden. Falls das TMS eine LogIn Nachricht mit dem Error Code 01 ablehnt, darf das RBC die Verbindung zum Zug nicht bestätigen.

{

"header": header,

"payload": {

"trainInfo": trainInfo

}

}

Listing 9: Log In JSON Format

## LogOut

Wenn ein Zug den Bereich, der von einem RBC überwacht wird, verlässt oder das Fahrzeug die End of Mission erreicht und abgerüstet wird, schickt das jeweilige RBC eine LogOut Nachricht an das TMS.

Falls ein Zug auf eine Verbindung zu einem anderen RBC wechseln möchte wird über eine LogOut Nachricht des einen RBC und eine [LogIn](#_LogIn) Nachricht des jeweils anderen RBC dieser Wechsel auch an das TMS gemeldet.

Diese Nachricht wird vom TMS mit einer [Error](#_Error) Nachricht quittiert. Eine LogOut Nachricht darf vom TMS nicht abgelehnt werden.

{

"header": header,

"payload": {

"trainInfo ": trainInfo

}

}

Listing 10: Log Out JSON Format

## Time Factor Changed

Wird die AppTime modifiziert (AppTime angehalten, neuer Faktor gesetzt), muss diese Änderung von einem RBC, das innerhalb der ETCS Applikation läuft und auf die AppTime zugreifen kann, an das TMS gemeldet werden. Dazu werden aktualisierte Werte für **„time\_base“** und „**time\_factor**“ in der Payload übermittelt.

Die Nachricht wird vom TMS mit einer entsprechenden [Error](#_Error) Nachricht quittiert. Eine Änderung darf nicht abgelehnt werden. (Es sei denn, die TimeChange Nachricht war ungültig.)

{

"header": header,

"payload": {

"time\_base": long | null,

"time\_factor": int

}

}

Listing 10: Log Out JSON Format

## TrainData

Bei Erzeugung oder Änderung der im Zug hinterlegten Validated Train Data [SRS TrainPacket 11] Informationen wird eine ETCS Message an das verbundene RBC gesendet. Dieses leitet die Informationen an das TMS in einer [TrainData](#_TrainData) Nachricht weiter.

{

"header": header,

"payload": {

"trainInfo": trainInfo,

"positionInfo": positionInfo,

"trainData": trainData

}

}

Listing 11: Train Data JSON Format

## Position Report

Ein Zug sendet zwar bei fast jeder Kommunikation mit dem RBC seine aktuelle Position mit. Es gibt jedoch eine weitere dedizierte Nachricht, um einen Position Report zu senden. Dieser wird nach den Parametern, die das RBC an den Zug zuvor übermittelt hat, zu bestimmten Zeitpunkten gesendet.

Allerdings können das TMS und das RBC einen Position Report auch zu einem Zeitpunkt sofort anfragen. Dafür nutzt das TMS die [Position Report Request](#_Position_Report_Request) Nachricht.

Die Position Report Nachricht enthält sowohl die Daten über den Zug (angegeben als trainInfo) als auch die Positionsinformationen des Zuges (angegeben als [positionInfo](#_PositionPacket)).

{

"header": header,

"payload": {

"trainInfo": trainInfo,

"positionInfo": positionInfo

}

}

Listing 12: Position Report JSON Format

## MA Request

Ein Zug sendet zu gewissen Zeitpunkten, definiert durch die MA Request Parameter, die das RBC an den Zug gesendet hat, eine MA Request Message an das RBC. Diese wird als MA Request Nachricht an das TMS weitergeleitet. Das TMS quittiert den Empfang mit einer [Error](#_Error) Nachricht mithilfe entsprechender Error Codes. Wurde die MA Request akzeptiert wartet das RBC nun auf eine [MA](#_MA) Nachricht vom TMS, die es dann an den Zug weiterleiten kann. Falls die MA Request abgelehnt wird, wird keine MA an den Zug gesendet, bis dieser eine neue MA anfragt oder das TMS von sich aus eine MA an den Zug schicken möchte. Demnach wartet das RBC nicht auf eine [MA](#_MA) Nachricht und die Konversation wird beendet.

Die MA Request Nachricht enthält die Informationen über den Zug und dessen Positionsangaben. Weiterhin enthält es eine Angabe über den Grund des MA Request. Dieser Wert wird im [SRS 7] definiert und als number angegeben.

{

"header": header,

"payload": {

"trainInfo": trainInfo,

"positionInfo": positionInfo,

"q\_marqstreason": int

}

}

Listing 13: MA Request JSON Format

## SH Request

Ein Zug kann beim RBC anfragen, ob das Fahrzeug in den SH Modus wechseln darf. Diese Anfrage wird zum TMS weitergeleitet, welches die Anfrage mit einer Error Nachricht akzeptiert oder ablehnt mithilfe der entsprechenden Error Codes.

Die SH Request Nachricht enthält dieselben Informationen wie eine [Position Report](#_Position_Report) Nachricht.

{

"header": header,

"payload": {

"trainInfo": trainInfo,

"positionInfo": positionInfo

}

}

Listing 14: SH Request JSON Format

# TMS Nachrichten

## Position Report Request

Ein Position Report von einem bestimmten Zug kann von dem TMS mit einer Position Report Request Nachricht angefragt werden. Das RBC quittiert den Empfang mit einer [Error](#_Error) Nachricht mit entsprechendem Error Code. Daraufhin fragt das RBC am jeweiligen Zug einen Position Report an und leitet diesen in einer [Position Report](#_Position_Report) Nachricht an das TMS weiter.

Eine Position Report Request Nachricht beinhaltet nur die eindeutige Fahrzeugnummer („nid\_engine“).

{

"header": header,

"payl": {

"nid\_engine": int

}

}

Listing 15: Position Report Request JSON Format

## MA

Eine MA Nachricht wird vom TMS aufgrund einer empfangenen [MA Request](#_MA_Request) Nachricht oder aus einem internen Grund gesendet. Das RBC leitet diese dann an den Zug weiter. Das RBC quittiert den Empfang der MA am Zug mit einer [Error](#_Error) Nachricht und dem entsprechenden Error Code.

Eine MA Nachricht enthält alle Informationen, die nach der ETCS Spezifikation in [SRS 7] und [SRS 8] zur Generierung einer MA Message [SRS 8 TrackMessage 3] oder einer MA Message with Shifted Location Reference [SRS 8 TrackMessage 33] notwendig sind. In der MA Nachricht können weitere optionale Informationen wie das „speedProfile“ [SRS 7 TrackPacket 21], das „gradientProfile“ [SRS 7 TrackPacket 27] oder das „modeProfile“ [SRS 7 TrackPacket 80] angegeben werden. Die Wertebereiche sind in [SRS 7] definiert und die Werte werden hier als number angegeben. Manche Werte können den Wert null annehmen.

{

"header": header,

"payload": {

"nid\_engine ": int,

"ma": {

"m\_ack": boolean,

"nid\_lrbg": int,

"q\_dir": int, // Could be overwritten in inner scope

"q\_scale": int, // Could be overwritten in inner scope

"d\_ref": int | null,

"eoa": {

"q\_dir": int | null,

"q\_scale": int | null,

"v\_loa": int,

"t\_loa": int,

"sections": [ // Min 1 Element

{

"l\_section": int,

"q\_sectiontimer": boolean,

"t\_sectiontimer": int | null,

"d\_sectiontimerstoploc": int | null

}

],

"endTimer": null | {

"t\_endtimer": int,

"d\_endtimerstartloc": int

},

"dangerPoint": null | {

"d\_dp": int,

"v\_releasedp": int

},

"overlap": null | {

"d\_startol": int,

"t\_ol": int,

"d\_ol": int,

"v\_releaseol": int

}

},

"gradientProfile": null | {

"q\_dir": int | null,

"q\_scale": int | null,

"gradients": [ // Min 1 Element

{

"d\_gradient": int,

"q\_gdir": boolean,

"g\_a": int

}

]

},

"speedProfile": null | {

"q\_dir": int | null,

"q\_scale": int | null,

"sections": [ // Min 1 Element

{

"d\_static": int,

"v\_static": int,

"q\_front": boolean,

"categories": [ // Min 1 Element

{

"q\_diff": int,

"nc\_cddiff": int,

"nc\_diff": int,

"v\_diff": int

}

]

}

]

},

"modeProfile": null | {

"q\_scale": int | null,

"modes": [ // Min 1 Element

{

"d\_mamode": int,

"m\_mamode": int,

"v\_mamode": int,

"l\_mamode": int,

"l\_ackmamode": int,

"q\_mamode": boolean

}

]

},

"linkingProfile": {

"q\_dir": int | null,

"q\_scale": int | null,

"links ": [{

"d\_link": int,

"nid\_c": int | null,

"nid\_bg": int,

"q\_linkorientation": boolean,

"q\_linkreaction": int,

"q\_locacc": int

}]

}

}

}

}

Listing 16: Movement Authority JSON Format

## Request to Shorten MA

Falls eine schon an den Zug gesendete MA nicht mehr gültig ist und daher verkürzt werden muss, schickt das TMS eine Anfrage inklusive einer neuen MA an das RBC in einer Request to Shorten MA Nachricht.

Der Inhalt der MA Informationen unterscheidet sich minimal zu denen in einer [MA](#_MA) Nachricht. Insbesondere werden keine Gradienten- oder Geschwindigkeitsprofile übertragen und es kann keine „d\_ref“ Information angegeben werden.

{

"header": header,

"payload": {

"nid\_engine": int,

"ma": {

"m\_ack": boolean,

"nid\_lrbg": int,

"q\_dir": int, // Could be overwritten in inner scope

"q\_scale": int, // Could be overwritten in inner scope

"eoa": {

"q\_dir": int | null,

"q\_scale": int | null,

"v\_loa": int,

"t\_loa": int,

"sections": [ // Min 1 Element

{

"l\_section": int,

"q\_sectiontimer": boolean,

"t\_sectiontimer": int | null,

"d\_sectiontimerstoploc": int | null

}

],

"endTimer": null | {

"t\_endtimer": int | null,

"d\_endtimerstartloc": int | null

},

"dangerPoint": null | {

"d\_dp": int,

"v\_releasedp": int

},

"overlap": null | {

"d\_startol": int,

"t\_ol": int,

"d\_ol": int,

"v\_releaseol": int

}

},

"modeProfile": null | {

"q\_scale": int,

"modes": [

{

"d\_mamode": int,

"m\_mamode": int,

"v\_mamode": int,

"l\_mamode": int,

"l\_ackmamode": int,

"q\_mamode": boolean

}

]

}

}

}

}

Listing 17: Request to Shorten MA JSON Format

## Conditional Emergency Stop

Das TMS kann einen bedingten Nothalt veranlassen. Dazu sendet es dem RBC eine Conditional Emergency Stop Nachricht. Der nötige Inhalt ist in der [SRS 8 TrackMessage 15] definiert.

{

"header": header,

"payload": {

"nid\_engine": int,

"nid\_em": int,

"m\_ack": boolean,

"q\_dir": int,

"q\_scale": int,

"nid\_lrbg": int,

"d\_ref": int,

"d\_emergencystop": int

}

}

Listing 18: Conditional Emergency Stop JSON Format

## Unconditional Emergency Stop

Das TMS kann einen unbedingten Nothalt veranlassen. Dazu sendet es dem RBC eine Unconditional Emergency Stop Nachricht. Der nötige Inhalt ist in der [SRS 8 TrackMessage 16] definiert.

{

"header": header,

"payload": {

"nid\_engine": int,

"nid\_em": int,

"m\_ack": boolean,

"nid\_lrbg": int,

}

}

Listing 19: Unconditional Emergency Stop JSON Format

## Revocation of Emergency Stop

Um einen Nothalt wieder aufzuheben kann das TMS mit Bezug auf die ursprüngliche ID der Nothalt Message „nid\_em“ (nicht „uuid“), eine Revocation of Emergency Stop Nachricht an das RBC senden.

Diese Nachricht enhält die nötigen Informationen nach [SRS 8 TrackMessage 18].

{

"header": header,

"payload": {

"nid\_engine": int,

"nid\_em": int,

"m\_ack": boolean,

"nid\_lrbg": int,

}

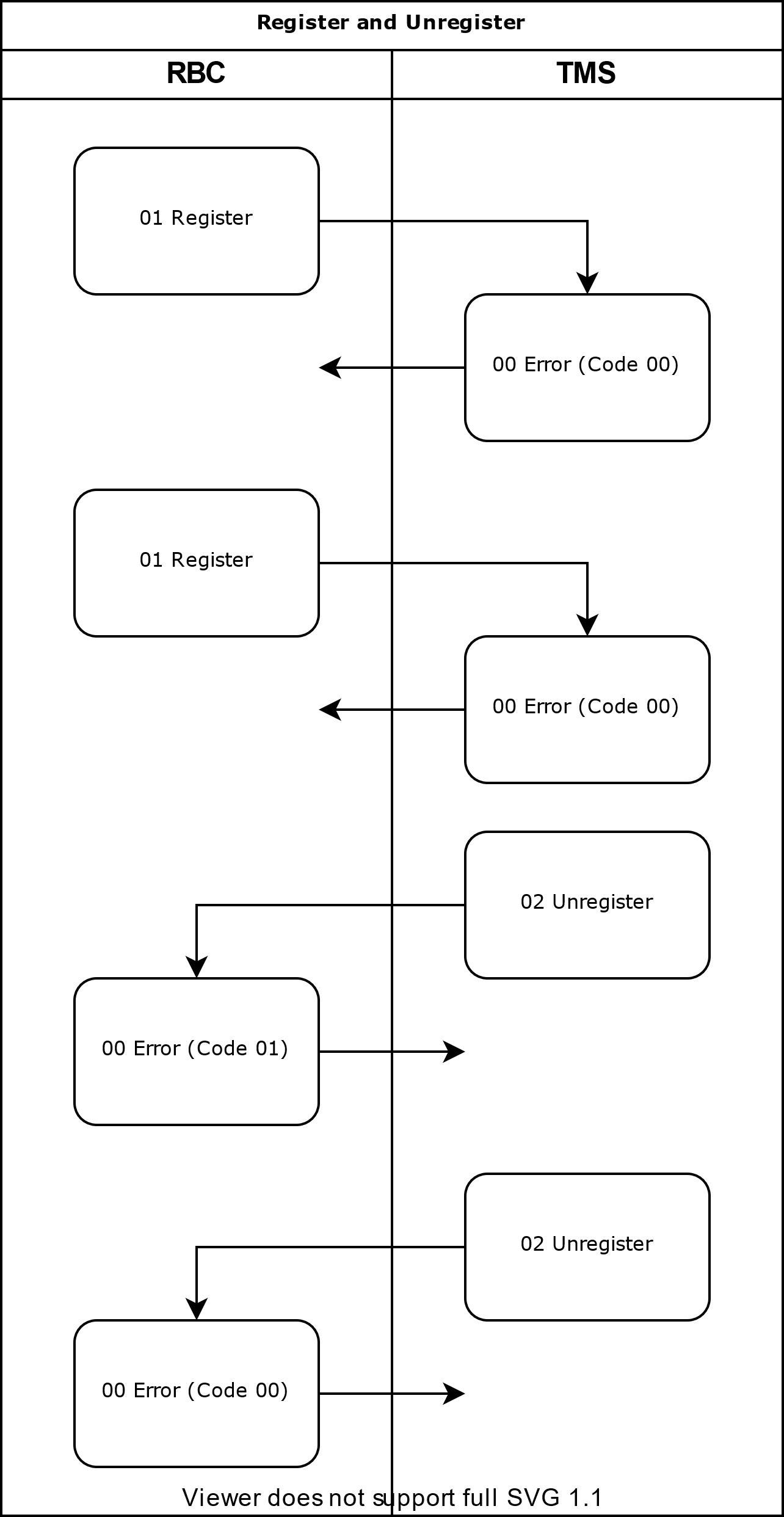
}

Listing 20: Revocation of Emergency Stop JSON Format

# Kommunikationsabläufe

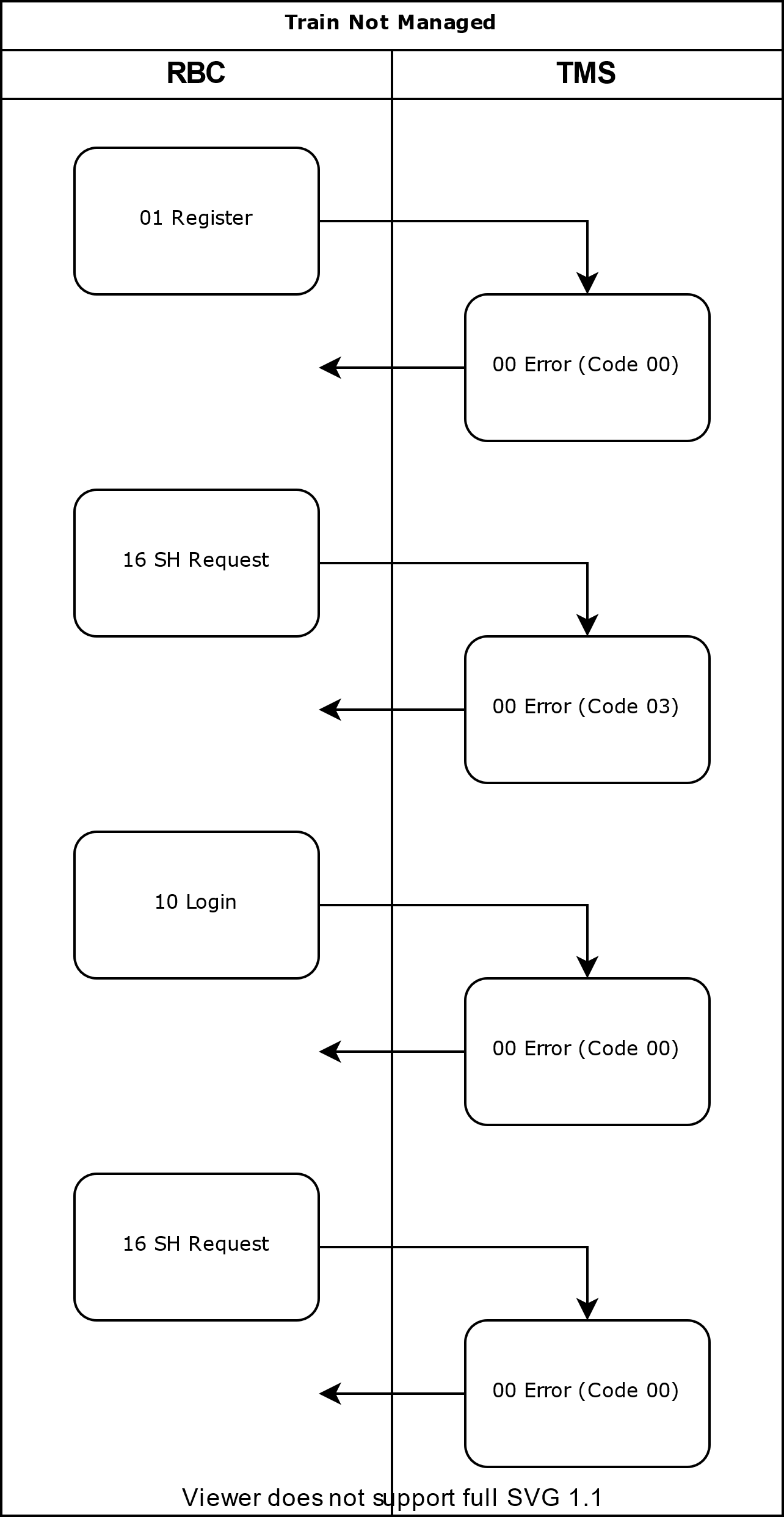
## Register und Unregister

Die Abbildung zeigt das Verhalten von den Nachrichten Register und Unregister. Im ersten Fall registriert sich das RBC am TMS. Da dies schon passiert ist schickt das TMS auch im zweiten Fall den Code 00 zurück. Im dritten Fall ist seitens des RBC eine laufende Konversation mit dem TMS noch nicht beendet (beispielsweise wird gerade auf eine Antwort auf eine MA Request gewartet). Demnach schickt das RBC den Error Code 01. Im letzten Fall erlaubt das RBC die Unregister Anfrage.



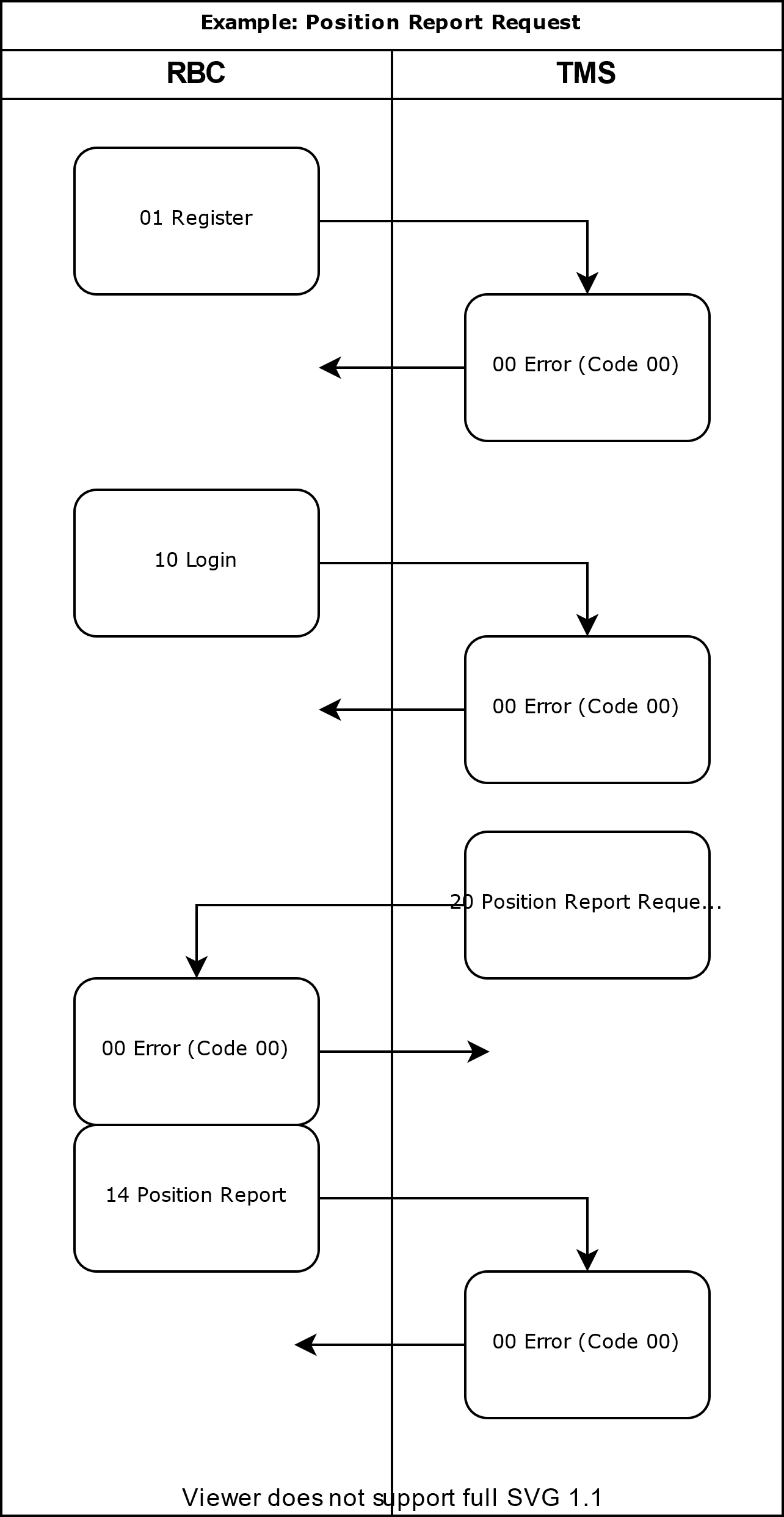
## Train Not Managed

Dies ist ein Beispiel für den Fall, dass ein Zug auf der TMS Seite noch nicht angemeldet wurde.



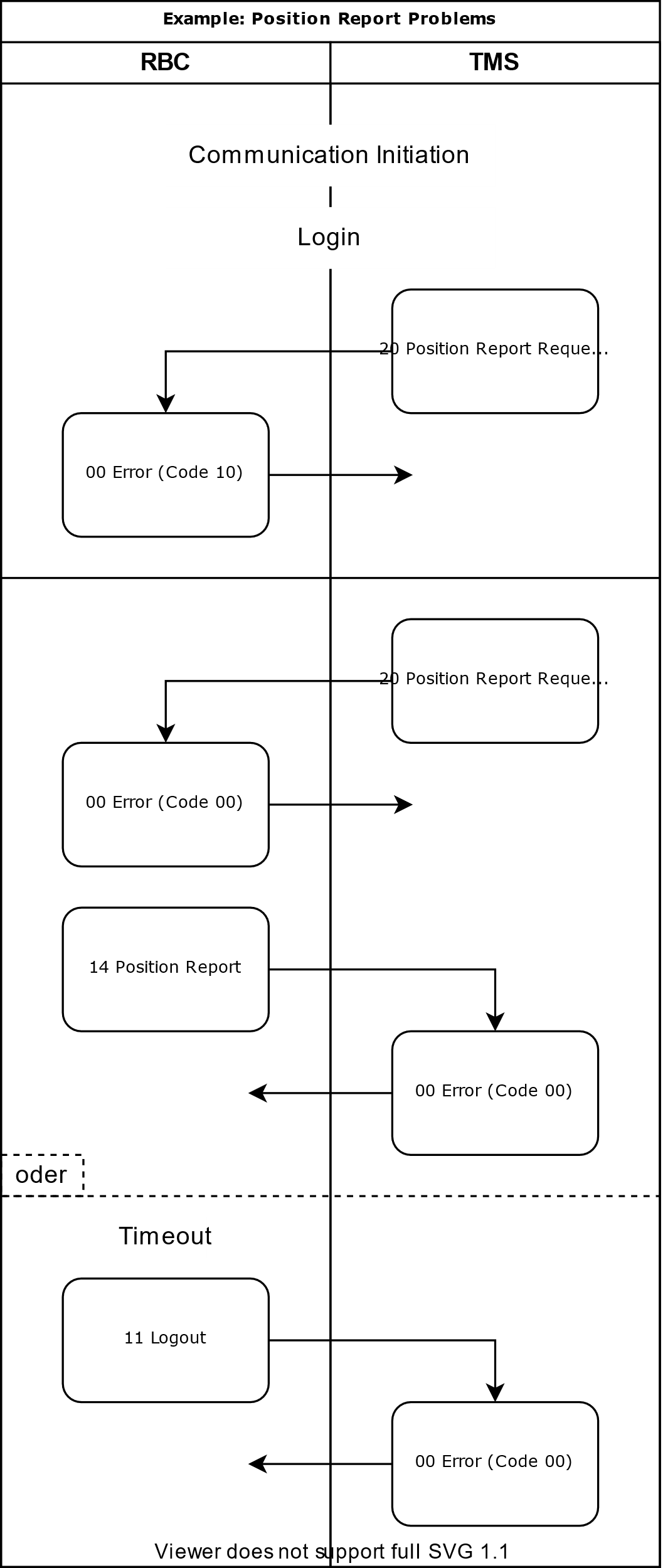
## Position Report Request

Dies ist ein Beispiel für eine Konversation zu einer Position Report Request Nachricht.



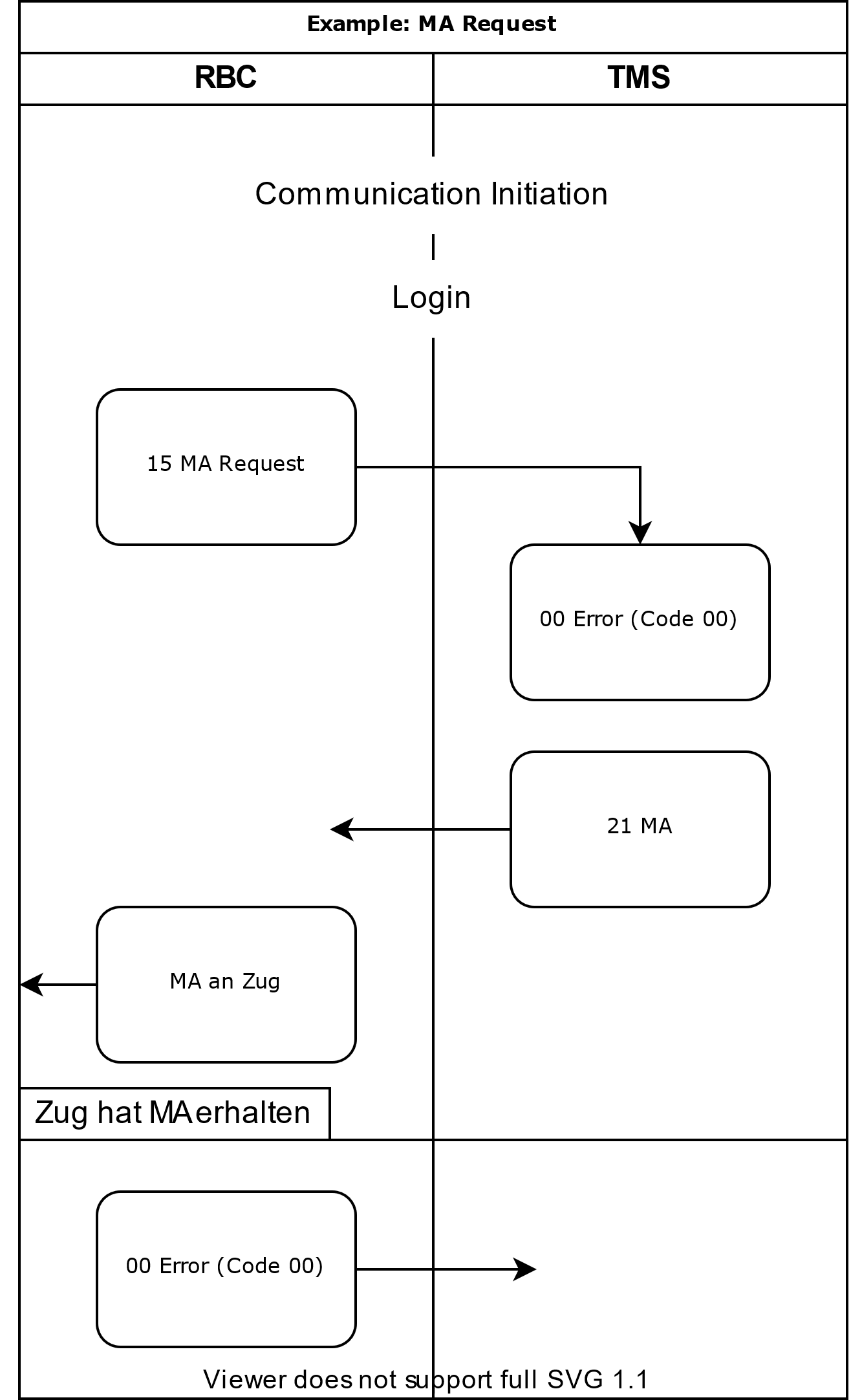
## Probleme beim Position Report

Es kann vorkommen, dass bei der Verbindung zum Zug Probleme auftauchen. Hier ist ein solcher Fall geschildert. Ist die Verbindung unterbrochen schickt das RBC den Error Code 10. Nach einiger Zeit kann das TMS seine Anfrage wiederholen. Kann die Verbindung zum Zug nicht mehr wieder hergestellt werden, meldet das RBC den Zug beim TMS wieder ab.



## MA Request

Hier ist ein Beispiel für das Senden einer MA Nachricht.



Verzeichnisse

Listingverzeichnis

[Listing 1: Message Type Definition 8](#_Toc35460706)

[Listing 2: Header Definition 9](#_Toc35460707)

[Listing 3: TrainInfo Definition 9](#_Toc35460708)

[Listing 4: PositionInfo Definition 10](#_Toc35460709)

[Listing 5: Error Type Definition 11](#_Toc35460710)

[Listing 6: Error JSON Format 11](#_Toc35460711)

[Listing 7: Register JSON Format 12](#_Toc35460712)

[Listing 8: Unregister JSON Format 12](#_Toc35460713)

[Listing 9: Log In JSON Format 13](#_Toc35460714)

[Listing 10: Log Out JSON Format 13](#_Toc35460715)

[Listing 11: Train Data JSON Format 14](#_Toc35460716)

[Listing 12: Position Report JSON Format 14](#_Toc35460717)

[Listing 13: MA Request JSON Format 15](#_Toc35460718)

[Listing 14: SH Request JSON Format 15](#_Toc35460719)

[Listing 15: Position Report Request JSON Format 16](#_Toc35460720)

[Listing 16: Movement Authority JSON Format 18](#_Toc35460721)

[Listing 17: Request to Shorten MA JSON Format 19](#_Toc35460722)

[Listing 18: Conditional Emergency Stop JSON Format 20](#_Toc35460723)

[Listing 19: Unconditional Emergency Stop JSON Format 20](#_Toc35460724)

[Listing 20: Revocation of Emergency Stop JSON Format 20](#_Toc35460725)

Literaturverzeichnis

*Das Bild der TU Darmstadt.* (23. 04 2019). Von https://www.intern.tu-darmstadt.de/media/medien\_stabsstelle\_km/services/medien\_cd/das\_bild\_der\_tu\_darmstadt.pdf abgerufen

*Das Corporate Design der TU Darmstadt*. (23. 04 2019). Von https://www.intern.tu-darmstadt.de/arbeitsmittel/corporate\_design\_vorlagen/index.de.jsp abgerufen

Glossar

|  |  |
| --- | --- |
| **DRAFT** | Der Status „DRAFT“ beschreibt die Entwicklungsphase eines Dokuments zur nächsten „FINAL“ Version. |
| **REVIEW** | Ein Dokument mit dem Status „REVIEW“ wird an eine weitere Person gegeben, die das Dokument auf inhaltliche und strukturelle Fehler überprüft. |
| **FINAL** | Das Dokument bekommt den Status „FINAL“ sobald die prüfende Person keine inhaltlichen oder strukturellen Fehler gefunden hat. |

Index

**Draft** 25

**Final** 25

**Review** 25

Anlagen

Anlage 1: Logos des ETCS Projekts

Originalgröße:





Größe für Titelseite von Dokumenten:





Symbol alleinstehend s/w (zur Verwendung in Präsentationsfolien):

