DISAG JSON Schnittstelle

V0.1	06.02.2018	CRO	Erste Fassung
V0.2	08.02.2018	CRO	OSS Einstellungen hinzugefügt
V0.3	14.02.2018	CRO	Definitionen Kommandos

Inhaltsverzeichnis

Allgemein, Sinn und Zweck

Die JSON Schnittstelle liefert in Echtzeit Informationen über die aktuelle Nutzung einer OpticScore-Anlage. Sie wird aus der OpticScoreServer-Software heraus bereitgestellt. Zudem werden Endgeräte über die Schnittstelle gesteuert, wenn eine parallele, gleichzeitige Ausführung von Befehlen auf mehreren Geräten notwendig ist.

OUT-JSONInterface.log

Die Log/Text-Datei wird unter %ProgramData%\DisagOpticScore erzeugt kontinuierlich beschrieben. Überschreitet die Datei eine Größe von 5MB, wird die aktuelle Datei nach OUT-JSONInterface.log.1 verschoben und OUT-JSONInterface.log weiter beschrieben. Es werden maximal zwei Dateien gehalten. In die Log-Datei werden nur Messages vom Typ „Event“ geschrieben.

Beim Öffnen der Datei muss darauf geachtet werden, dass diese nicht im exklusiven Zugriff geöffnet wird, da die OSS die Datei sonst nicht befüllen kann.

UDP Broadcast

Alle Informationen werden parallel zur Logdatei über einem UDP Broadcast auf Port 30169 im lokalen Netzwerk verteilt. Mit einem Client können die Daten auf diesem Port abgegriffen werden.

Beispiel-Client unter Java:

```java
package de.disag.jsoninterface;

import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class BroadcastListener {

    public static void main(String[] args) {
        int nPort = 30169;

        if (args.length > 0) {
            nPort = Integer.parseInt(args[0]);
        }

        try {
            DatagramSocket dsocket = new DatagramSocket(nPort);
            byte[] buffer = new byte[4096];

            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

            while (true) {
                dsocket.receive(packet);

                String msg = new String(buffer, 0, packet.getLength());
                System.out.println(packet.getAddress().getHostName() + ": " + msg);

                packet.setLength(buffer.length);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

Struktur der JSON Objekte

Alle Objekte besitzen das Attribut „UUID“. Dabei handelt es sich um eine UUIDv4 (uppercase), z. B.: 6727B7C5-D772-4419-ABCC-A84098F3C91C.

Message

Parameter	Typ	Beschreibung	Werte
MessageType	enum	MessageType	

MessageVerb	enum	**MessageVerb**	
Ranges	Command: Array<int> Event: int	Liste der Stände, die von einem Kommando betroffen sind oder Standnummer von dem ein Event ausgelöst wurde	
Sequential	bool	Wenn mehrere Kommandos in Objects vorhanden sind, bedeutet Sequential = true, dass diese Befehle sequentiell abgearbeitet werden müssen	
Objects	Array	z. B. Liste von **Shot**	

TLightSequence

Parameter	Typ	Beschreibung	Werte
DelayBeforeStart	long	Zeit in ms bevor die Sequenz gestartet werden soll	
FirstStatus	enum	**TrafficLightStatus**	
FirstStatusDuration	long	Dauer des FirstStatus in ms	
SecondStatus	enum	**TrafficLightStatus**	
SecondStatusDuration		Dauer des SecondStatus in ms	
Loops	int	Anzahl der Durchläufe der Sequenz	

Shot

Parameter	Typ	Beschreibung	Werte
ShotDateTime	DateTime	Zeitstempel des Schusses	yyyy-MM-dd HH:mm:ss.fff
TLStatus	enum	Zustand der Ampel während des Schusses, **TrafficLightStatus**	off, red, green
LastTLChange	int	Zeit in ms seit letzter Ampelschaltung	>= 0
Source	enum	Herkunft des Schusses; OpticScore = Messrahmen	OpticScore, RedDot
Range	int	Standnummer	>= 0
Shooter	object	**Shooter**	
DiscType	enum	**DiscType**	LG, LGA, LP, LPA, KK, KKA, KYFFH, ZS, ZSA, ZSTRD, LPS, LPI
X	int	X-Position des Schusses (vom Zentrum des Messbereichs aus)	-9000 < X < 9000
Y	int	Y-Position des Schusses	-9000 < Y < 9000
Distance	float	Teiler	0 <= Distance <= 12700
Count	int	Nummer des Schusses	>= 0
FullValue	int	Ringwert ohne Zehntel	0 <= FullValue <= 10
DecValue	float	Ringwert mit Zehntel	0 <= DecValue <= 10.9

Run	int	Durchgang innerhalb der Disziplin	>= 0
IsValid	bool	Schuss gültig	true, false
IsWarmup	bool	Probeschuss	true, false
IsHot	bool	Wertungsschuss	true, false
IsDummy	bool	Generierter Schuss	true, false
IsInnerten	bool	Innenzehner	true, false
IsShootoff	bool	Stechschuss	true, false
MenuItem	object	**MenuItem**	
Remark	string	Hinweis, z. B. Ringabzug	

Series

Parameter	Typ	Beschreibung	Werte
Shooter	object	Shooter	
ID	int	Nummer der Serie	ID > 0
FullValue	int	Serienwert ohne Zehntel	FullValue >= 0 <= (SeriesLength * 10)
DecValue	float	Serienwert mit Zehntel	DecValue >= 0 <= (SeriesLength * 10.9)
SeriesLength	int	Anzahl Schuss pro Serie	SeriesLength > 0, Default: 10

Result

Parameter	Typ	Beschreibung	Werte
Shooter	object	Shooter	
FullValue	int	Ergebnis ohne Zehntel	FullValue >= 0 <= (ShotCount * 10)
DecValue	float	Ergebnis mit Zehntel	DecValue >= 0 <= (ShotCount * 10.9)
ShotCount	int	Anzahl Schuss	SeriesLength > 0, Default: 10

Shooter

Parameter	Typ	Beschreibung	Werte
Firstname	string	Vorname	
Lastname	string	Nachname	
Birthyear	int	Geburtsjahr	>1900
InternalID	string	Interne ID	
Identification	string	Passnummer	
Team	object	**Team**	Kann NULL sein
Club	object	**Club**	

Club

Parameter	Typ	Beschreibung	Werte
Name	string	Vereinsname	
ShortName	string	Vereinsname (kurz)	

ID	string	Vereinsnummer	

Team

Parameter	Typ	Beschreibung	Werte
Name	string	Mannschaftsname	
ShortName	string	Mannschaftsname (kurz)	

MenuItem

Parameter	Typ	Beschreibung	Werte
MenuID	string	ID des Menüeintrags	z. B. 100_4
MenuPointName	string	Name des Menüpunkts	z. B. „Neue Schießzeiten“
MenuItemName	string	Name des Eintrags	z. B. „LG 40 Schuss“

RangeSettings

Parameter	Typ	Beschreibung	Werte
IsLocked	bool	Stand gesperrt	
ShowTenth	bool	Zehntel anzeigen	
ShowTenthSum	bool	Zehntelsumme anzeigen	
ShowDistance	bool	Teiler anzeigen	
ShowAverage	bool	Treffermittelpunkt anzeigen	
ShowPerimeter	bool	Umkreis anzeigen	

Competition

Parameter	Typ	Beschreibung	Werte
Shooter	object	**Shooter**	
MenuItem	object	**MenuItem**	
ResultID	string	UUIDv4	

CompetitionContext

Parameter	Typ	Beschreibung	Werte
CompetitionStatus	enum	**CompetitionStatus**	
RemainingTime	int	Verbleibende Zeit in Sekunden	

Enumerationen

MessageVerb

Wert	Command	Event
Various	Wenn mehrere verschiedene Befehle in Objects vorhanden hat jeder davon	

	hat den Parameter CommandType (entspricht **MessageType**) für sich gesetzt.	
Shot		Schuss gefallen
Series		Serie fertig
Result		Gesamtergebnis fertig
TLightSequence	Ampelsequenz ausführen	Ampel geschalten
RangeSettings	Einstellungen übernehmen	Einstellungen wurden geändert
Competition	Wettkampf vorbereiten	
Competition	Wettkampfeinstellungen übernehmen	Wettkampfeinstellungen wurden geändert

MessageType

Wert	Beschreibung
Command	Der Empfänger der Message muss prüfen, ob sie für ihn von Bedeutung ist (anhand der Ranges) und ggf. ausführen
Event	Die Message enthält Informationen wie Schüsse, Standbelegung, etc. Das Attribut Sequential ist uninteressant. Pro Event-Message ist genau ein Objekt in Objects Bei Events werden nur die Werte in den Objekten gesetzt, die sich geändert haben, z. B. wenn Zehntelschuss an einem Stand aktiviert wurde, werden nicht alle Einstellungen geliefert

DiscType

Wert	Beschreibung
LG	Luftgewehr
LGA	Luftgewehr-Auflage
LP	Luftpistole
LPA	Luftpistole-Auflage
KK	Kleinkalibergewehr
KKA	Kleinkalibergewehr-Auflage
ZS	Zimmerstutzen
ZSA	Zimmerstutzen-Auflage
ZSTRD	Zimmerstutzen-Traditionell
KYFFH	Kyffhäuserscheibe
LPS	DE Luftpistole Schnellfeuer
LPI	IT Luftpistole

TrafficLightStatus

Wert	Beschreibung
off	Ampel ist aus
red	Ampel ist rot

green	Ampel ist grün

CompetitionStatus

Wert	Beschreibung
startNextRun	Nächsten (oder ersten Run) starten
warmupStart	Probeschießen starten
hotStart	Wertungsschießen starten
pause	Aktuellen Run unterbrechen
resume	Aktuellen Run fortführen
warmupSelfStart	Start vorbereiten, Schütze startet am Stand selbstständig

Einstellungen in der OSS

Die Einstellungen der JSON Schnittstelle können in der OSS unter Extras -> Optionen -> „JSON Live“ vorgenommen werden.