

Version - ()

Document Information

|  |  |
| --- | --- |
| Version |  |
| Release Date |  |
| Author(s) |  |
| Approved By |  |

Document Log

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Summary of Changes |
| 0.1 | 25/07/2000 | Andrew Paull | Initial Draft |
| 0.2 | 06/02/2001 | Andrew Paull | Draft 2: Protocol version number added to the message header. (PROT = 0x01)  Customer Name added to the validation response. |
| 0.3 | 09/03/2001 |  | Draft 3: Error code added to the TransactionFailed message.  General Error response added to indicate if received message incorrectly formatted.  InvalidCard msg sent in response to a BalanceRequest for unknown/invalid card. |
| 1.0 | 27/08/2001 | Andrew Paull | CRC Generation Appendix added. |
| 2.0 | 14/10/2009 | Matt Reid | Added 5.2.7, 5.3.5, 5.3.6 and 5.3.7 |
| 3.0 | 3/2/202010 | Matt Reid | Added new Simplay functionality 5.28, 5.2.9, 5.38 and 5.39 |
| 3.1 | 25/02/2010 | Matt Reid | Added parts of points accrual and redemption 5.3.11, 5.3.12, 5.3.13, 5.3.14 |
| 4.0 | 27/04/2010 | Matt Reid | Added new Barrel draw support 5.2.10, 5.3.15, 5.3.16 |
|  | 27/04/2010 | Matt Reid | Added new EGM Position number query 5.2.11, 5.3.17 |
| 4.1 | 25/6/2010 | Matt Reid | 5.2.12 Jackpot win message |
| 4.2 | 28/6/2010 | Matt Reid | 5.2.13 Request player preferences |
|  |  | Matt Reid | 5.3.18 Player preference reply |
| 5.0 | 04/02/2011 | Matt Reid | Added new matrix for implementation details (Section 5) |
|  |  | Matt Reid | Added new interface to include spend amount (in cents) 6.3.9 |
|  |  | Matt Reid | Removed references to Jupiter’s and renamed to Maxgaming (throughout document) |
| 6.0 | 08/05/2012 | Trevor Miller | Moved to Technical Specifications Template |
|  |  | Trevor Miller | Added new section for PurchaseItemList message 1020H |
| 6.1 | 13/6/2012 | Declan Martschinke | Updated matrix to reflect status of 6.3.9 being in production |
| 6.2 | 18/6/2012 | Vinayak Kankanwadi | Updated section 6.4.20 PurchaseItemList 1020H to be XML Format. |
| 6.3 | 29/6/2012 | Vinayak Kankanwadi | Added section 9.3 and 9.4 to provide more information on Purchase Item message |
| 6.4 | 17/08/12 | Vinayak Kankanwadi | Added section 9.5 to provide more information on sending Sale Information more than Maximum limit |
| 6.5 | 03/09/12 | Vinayak Kankanwadi | Sections related to Message Flow and Database Schema moved to POS CPI Message Flow.doc and the same link is added as RD-2 |

Copyright

Copyright @ , ABN

All rights reserved

This document and all related documentation is the property of . Apart from any use as permitted under the Copyright Act 1968, no part may be reproduced by any process without written permission from .

**Table of Contents**

[1. Introduction 5](#_Toc334442926)

[1.1. Purpose 5](#_Toc334442927)

[1.2. Scope 5](#_Toc334442928)

[1.3. Objectives 5](#_Toc334442929)

[1.4. Audience 5](#_Toc334442930)

[1.5. Glossary of Terms 5](#_Toc334442931)

[1.6. Related Documents 5](#_Toc334442932)

[1.7. Questions / Issues Register 6](#_Toc334442933)

[2. Environmental Requirements 6](#_Toc334442934)

[2.1. Infrastructure Requirements 6](#_Toc334442935)

[2.2. Software Requirements 7](#_Toc334442936)

[2.3. Communications Requirements 7](#_Toc334442937)

[3. Development Overview 8](#_Toc334442938)

[3.1. Background 8](#_Toc334442939)

[4. Application Protocol 8](#_Toc334442940)

[4.1. Protocol Implementation Matrix 8](#_Toc334442941)

[4.2. Data Packet Format 9](#_Toc334442942)

[4.3. Messages sent from the Cougar Interface 10](#_Toc334442943)

[4.4. Messages sent from the POS 13](#_Toc334442944)

[5. Appendices 19](#_Toc334442945)

[5.1. Maxgaming Card Format 19](#_Toc334442946)

[5.2. 16 bit CRC Generation 19](#_Toc334442947)

[5.3. Purchase Item Sale XML Node’s maximum length 20](#_Toc334442948)

[5.4. Purchase Item Sale XML content sample 21](#_Toc334442949)

[5.5. Send Information more than the limit 21](#_Toc334442950)

# Introduction

## Purpose

This document defines the interface protocol that the Cougar monitoring system uses to communicate with 3rd party POS systems. It is intended for use by both internal and external developers.

## Scope

### In Scope

* Protocol Messages sent to a 3rd Party Point of Sale (POS) system
* Protocol Messages received by 3rd Party Point of Sale (POS) system

### Out of Scope

* Reports, dashboards, or any analsys of the data that these message produce within the Cougar Monitoring system

## Objectives

Maxgaming Cougar monitoring system contains a player loyalty system that enables a player to accrue loyalty points through gaming machine play for redemption of goods or services at a later time.

This document defines the protocol between the Cougar POS Interface (CPI) and 3rd party POS systems that may be used to accrue/redeem points. A player may either earn points via the purchase of food and beverages at a POS or pay for the same services using their point balance.

This interface gives the POS system access to the current points balance stored in the Cougar system and allows the POS to accrue to / redeem from this balance.

## Audience

Maxgaming Technical / Development Staff

3rd Party Point of Sale Technical / Development Staff

## Glossary of Terms

|  |  |
| --- | --- |
| Term | Description |
| CPI | Cougar POS Interface |
| LAN | Local Area Network |
| POS | Point of Sale |

## 

## Related Documents

|  |  |
| --- | --- |
| Ref | Description |
|  | [Point of Sale Emulator](https://tattscentral.tattsgroup.com/sites/technology/maxgaming/Maxgaming%20Documents/POS%20Emulator.zip) |
|  | [POS CPI Message Flow and Database Schema](https://tattscentral.tattsgroup.com/sites/technology/maxgaming/Maxgaming%20Documents/POS%20CPI%20Message%20Flow.doc) |

## Questions / Issues Register

|  |  |  |
| --- | --- | --- |
| Ref | Status | Description |
|  | Open | Loading this level of detail into the Venue / Host system is going to expand the databases (Host/Venue) significantly, the risk of hardware (disk) updates needs to be acknowledged by the business, ensure that this risk/outcome has been provisioned for and that the initial sites are monitored closely |
|  | Open | Moving this data from venue to host may require some sort of QOS in data transaction from venue to host or a venue based component which schedules the upload of POS data from venue to host ensure that the latency of core monitoring alerts/data between venue/host is not compromised by this sudden increase in data traffic (refer to new proposed venue Component). |

# Environmental Requirements

## Infrastructure Requirements

The infrastructure required to operate this function is the venue’s site controller. The is a PC running a Windows Operating System. The PC must have an network (Ethernet) connection to the Venue’s Gaming LAN.

* Point of Sale Terminal
* Cougar Site Controller PC
* Cabling of POS Server / Terminal into the Gaming Network (Ethernet)



## Software Requirements

The software required to run this interface with the Cougar Monitoring System is the CPI (Cougar Point of Sale Interface), this is installed on the Cougar Site Controller.

Cougar Site Controller

* Windows Operating System
* SQL Server database
* Cougar Monitoring System Components (CPI)

## Communications Requirements

The communications from the Cougar system to the POS system may be over various media. The application data transmitted will be the same, however differing transport layer packaging may be utilised for the various communications media. The supported communication technologies are listed below

### Serial Communications

Data will be transmitted over an RS232 or RS485 serial line with the following base settings:

* Data rate 9600 baud
* 1 Stop bit
* 8 data bits
* No parity
* No handshaking

Data transmitted over the serial line will be packeted and take the format:

<STX><Data><ETX>

* <STX> - start of transmission Byte, code 03 Hex
* <Data> - the application layer message
* <ETX> - the end of transmission byte, code 04 Hex

If an STX is received before an ETX then any previous data will be discarded and a new data packet started.

On transmission, the application data must be encoded with an escape code 10 Hex to allow transmission of data corresponding to control characters. To encode the application data prior to transmission the data must be scanned and escape characters inserted for data bytes less than or equal to the escape character. When an escape character is used, the following byte (the escaped byte) has the escape code added to its value. The application data must be fully assembled before the data is encoded.

To decode the data, the application data is scanned and any escape characters are removed from the data stream and the byte following the escape character has the value of the escape character subtracted from it. The escape characters must be removed before evaluating the application data;

Eg. <03><FF><10><13><04> translates to

<03><FF><03><04>

### TCP/IP Communications

The site controller is able to communicate to remote POS systems over a LAN using TCP/IP communications. The POS listens on socket 8005 hex and the site controller connects to the socket, transmitting and receiving data over the socket connection. There are no additional structures placed around the application data packets when using TCP/IP communications.

# Development Overview

## Background

The interface has been originally created to allow the Maxgaming Loyalty system greater access to events which “may” accrue loyalty points. Within the Cougar monitoring system we can ascertain specific gaming information (turnover, jackpot wins, etc…) can be converted to Member Loyalty points. With this interface other functions of the venue (Food, Beverage, etc…) can also be integrated into the accumulation & redemption of loyalty points.

# Application Protocol

## Protocol Implementation Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Development Complete | Emulators Available | Production Approved |
| BalanceRequest 1001H | Yes | Yes | Yes |
| AccruePoints 1002H | Yes | Yes | Yes |
| RedeemPoints 1003H | Yes | Yes | Yes |
| TransactionStatus 1004H | Yes | Yes | Yes |
| MemberInfoRequest 1005H | Yes | Yes | Yes |
| AccruePointsWithType 1006H | Yes | Yes | Yes |
| RedeemPointsWithType 1007H | Yes | Yes | Yes |
| CashBalanceRequest 1008H | Yes | Yes | Yes |
| CashPurchaseRequest 1009H | Yes | Yes | Yes |
| AccruePartPoints 1011H | Yes | Yes | Yes |
| RedeempPartPoints 1012H | Yes | Yes | Yes |
| AccruePartPointsWithType 1013H | Yes | Yes | Yes |
| RedeemPartPointsWithType 1014H | Yes | Yes | Yes |
| RequestBarrelDrawList 1015H | No | No | No |
| PrintBarrelDrawTicket 1016H | No | No | No |
| RequestCardOnEGMPosition 1017H | Yes | Yes | Yes |
| PlayerPreferenceReply 1018H | No | No | No |
| TransactionDollarValue 1019H | Yes | Yes | Yes |
| PurchaseItemList 1020H | No | No | No |

## Data Packet Format

The application data will be transmitted using data packets. The data packet will have the structure:

<LHI><LLO><FHI><FLO><PROT><ADATA><CRCHI><CRCLO>

* LHI,LLO – 2 byte length (in bytes) of data packet, including the length bytes and excluding the 2 byte CRC
* FHI,FLO – 2 byte function code of this packet
* PROT – 1 byte protocol version number. Currently set to 0x01.
* ADATA – 0..n bytes of additional data, dependent on packet function
* CRCHI,CRCLO – 2 byte, 16 bit CRC of packet, 0 seeded. See Appendix A.

Note ☞ All string data is sent using standard single byte ASCII characters

16 bit CCITT CRC of the message utilizing the standard polynomial, X16 +X12 +X5 +1. Seed values are always 0 (zero). {‘C’ source available on request}

If data is received by the POS system and the CRC is not correct the POS system will ignore the data.

## Messages sent from the Cougar Interface

### PointBalance – function 0001H

Contains the point balance for a player.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<AMNT> - 32 bit (MSB first) Current Point Balance

<NAME 0..30> - ASCII text containing the player’s name

### TransactionSuccessful – function 0002H

The last transaction for the player was successful. The response contains the transaction number received from the POS.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<TXN\_ID> - 32 bit (MSB first) Supplied by the POS.

### TransactionFailed – function 0003H

The last transaction for the player failed. The response contains the transaction number received from the POS.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<TXN\_ID> - 32 bit (MSB first) Supplied by the POS.

<ERROR> - 8 bit error code

0x01 – Invalid Card

0x02 – Duplicate Transaction

0x03 – General Transaction Error

0x04 – Invalid Point Type

0x05 – Module not implemented

0x06 – Insufficient Cash Funds

0x07 – Exceeded Cash Account Limit

0x08 – Exceeded Cash POS Purchase Limit

0x09 – Terminal ID not specified

0x10 – Invalid parameters

### TransactionStatus – function 0004H

The response contains the status of the transaction.

Additional Data:

<TXN\_ID> - 32 bit (MSB first)

<STATUS> - 8 bit Transaction Status

0x01 – Successful

0x02 – Failed

0x03 – In progress

0x04 – Unknown transaction

### Invalid Card – function 0005H

Sent in response to Balance Request if the card is unknown / invalid.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

### General Error – function 0006H

Sent to the POS system in response to received message with invalid data in the message header.

Additional Data:

<ERROR> - 8 bit Transaction Status

0x01 – Invalid message CRC

0x02 – Invalid message length

0x03 – Unknown function code

0x04 – Invalid Packet (missing STX, ETX)

0x05 – System Unavailable

0x06 – Module not implemented

### MemberInfo – function 0007H

Contains the point balance and membership details for a player.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<MEMBERSHIPID> - 32 bit (MSB first) Membership ID

<RATINGGRADE> - 32 bit (MSB first) Rating Grade ID

<MEMBERTYPE> - 32 bit (MSB first) Membership Type ID

<POINTS> - 8 bit Number of points and balances

Repeated for 1..POINTS

<POINTTYPEID> - 32 bit (MSB first) Point Type ID

<AMNT> - 32 bit (MSB first) Current Point Balance

<POINTNAME> - 30 BYTES – 8 bit ASCII text string describing point type

End Repeat

<NAME 0..30> - ASCII text containing the player’s name

### CashBalanceReply – function 0008H

Contains the Cash balance and POS limit for a site.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<Amount> - 32 bit (MSB) Amount in cents

<Limit> - 32 bit (MSB) POS Limit for single purchases in cents

### CashTransactionSucessful – function 0009H

The last transaction for the player was successful. The response contains the transaction number received from the POS and balance remaining.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<TXN\_ID> - 32 bit (MSB first)

<Amount> - 32 bit (MSB first) Amount in cents of purchase

<Balance remaining> - 32 bit (MSB) Cash balance remaining in cents

<Receipt ID> - 32 bit (MSB first) Maxgaming internal receipt number

### BarrelDrawListReply – function 0010H

The list of barrel draws currently enabled at the venue sent in response to a RequestBarrelDrawList – function 1015H transaction from the POS. If there are no barrel draws set up at the venue, then the count will return 0.

Additional Data:

<COUNT> - 8 bit Number of barrel draw items configured at the site

Repeated for 1..COUNT

< BARRELDRAWID > - 32 bit (MSB first) Barrel Draw ID.

<NAME> - 30 BYTES – 8 bit ASCII text string describing barrel draw promotion

End Repeat

### CardOnEGMPositionReply – function 0011H

Sent in response to a RequestCardOnEGMPosition – function 1017H request and returns card ID of 0 if no player on EGM position.

Additional Data:

<POSITION> - 32 bit (MSB first) EGM Position Number

<CARD> - 32 bit (MSB first) Card ID

### JackpotWin – function 0012H

This is sent when a megapoints Jackpot is won due to turnover in the most recent POS transaction.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<WinPoints> - 32 bit (MSB) Jackpot win amount (in points)

### RequestPlayerPreferences – function 0013H

This is sent by the Maxgaming system to request the most purchased item(s) of a particular type. The categories are: 1) Drink 2) Meal 3) Other. The Maxgaming system can request a number of the most of the frequent purchases as well, specified in the Num field. For example to request the 3 most common drink purchases for Card ID 600900: <600900><1><3>

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<Category> - 8 bit Category type ID as above

<Num> - 8 bit Number

## Messages sent from the POS

### BalanceRequest – function 1001H

Sent to request the point balance for the player.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

### AccruePoints– function 1002H

Sent to request that <AMNT> points be added to the player’s balance.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<AMNT> - 32 bit (MSB first) Accrual amount

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### RedeemPoints– function 1003H

Sent to request that <AMNT> points be withdrawn from the player’s balance.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<AMNT> - 32 bit (MSB first) Redemption amount

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### TransactionStatus – function 1004H

Sent to request the status of the transaction.

Additional Data:

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

Note:

In order not to conflict with internal transaction ID’s used in the Cougar system all ID’s must be less than 106 and be unique.

### MemberInfoRequest – function 1005H

Sent to request the point balance, rating, membership and member type for the player (see 5.2.7). If site has not requested this functionality a General Error packet (0x0006) with error code 0x06 is returned.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

### AccruePointsWithType– function 1006H

Sent to request that <AMNT> points be added to the player’s balance on a specified point type. If site has not requested this functionality a Transaction Failed packet (0x0003) with error code 0x05 is returned.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<POINTTYPEID> 32 bit (MSB first) Point Type ID.

<AMNT> - 32 bit (MSB first) Accrual amount

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### RedeemPointsWithType– function 1007H

Sent to request that <AMNT> points be withdrawn from the player’s balance on a specified point type. If site has not requested this functionality a Transaction Failed packet (0x0003) with error code 0x05 is returned.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<POINTTYPEID> 32 bit (MSB first) Point Type ID.

<AMNT> - 32 bit (MSB first) Redemption amount

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### CashBalanceRequest – function 1008H

Sent to request the Cash balance for the player.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

### CashPurchaseRequest – function 1009H

Sent to request a Cash purchase for the player.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<Amount> - 32 bit (MSB) Amount in cents

<Terminal ID 0..15> - ASCII text of terminal ID, leading zeros (e.g. “000000Till12345”

<DESC 0…25> - ASCII text describing the POS transaction. E.g. “Drinks Purchase”

### CashDepositRequest – function 1010H

Sent to request a Cash deposit for the player.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<Amount> - 32 bit (MSB) Amount in cents

<Terminal ID 0..15> - ASCII text of terminal ID, leading zeros (e.g. “000000Till12345”

<DESC 0…25> - ASCII text describing the POS transaction. E.g. “Drinks Purchase”

### AccruePartPoints– function 1011H

Sent to request that <AMNT> points be added to the player’s balance.

To send an accrual for 8.25 points send 8 in the <AMNT> field and 25 in the <PART> field.

Will return transaction failed with an error code of 0x10 (invalid parameters) if <PART> is > 100.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<AMNT> - 32 bit (MSB first) Accrual amount

<PART> - 8 bit Decimal part of point 0…100

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### RedeemPartPoints– function 1012H

Sent to request that <AMNT> points be withdrawn from the player’s balance.

To send redemption for 8.25 points send 8 in the <AMNT> field and 25 in the <PART> field.

Will return transaction failed with an error code of 0x10 (invalid parameters) if <PART> is > 100.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<AMNT> - 32 bit (MSB first) Redemption amount

<PART> - 8 bit Decimal part of point 0…100

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### AccruePartPointsWithType– function 1013H

Sent to request that <AMNT> points be added to the player’s balance on a specified point type. If site has not requested this functionality a Transaction Failed packet (0x0003) with error code 0x05 is returned.

To send an accrual for 8.25 points send 8 in the <AMNT> field and 25 in the <PART> field.

Will return transaction failed with an error code of 0x10 (invalid parameters) if <PART> is > 100.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<POINTTYPEID> 32 bit (MSB first) Point Type ID.

<AMNT> - 32 bit (MSB first) Accrual amount

<PART> - 8 bit Decimal part of point 0…100

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### RedeemPartPointsWithType– function 1014H

Sent to request that <AMNT> points be withdrawn from the player’s balance on a specified point type. If site has not requested this functionality a Transaction Failed packet (0x0003) with error code 0x05 is returned.

To send redemption for 8.25 points send 8 in the <AMNT> field and 25 in the <PART> field.

Will return transaction failed with an error code of 0x10 (invalid parameters) if <PART> is > 100.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<POINTTYPEID> 32 bit (MSB first) Point Type ID.

<AMNT> - 32 bit (MSB first) Redemption amount

<PART> - 8 bit Decimal part of point 0…100

<DESC 0..25> - ASCII text describing the POS transaction. Eg “Drinks Purchase”

### RequestBarrelDrawList – function 1015H

Sent to request all the Barrel Draws configured at the site

Additional Data:

No additional data

### PrintBarrelDrawTicket – function 1016H

Sent to request a barrel draw ticket be printed for a particular Card ID

Additional Data:

<CARD> - 32 bit (MSB first) Card ID (as per CRD field in track1 format)

< BARRELDRAWID > - 32 bit (MSB first) Barrel Draw ID.

<NUM> - 8 bit (MSB first) Number of barrel draw tickets to print

### RequestCardOnEGMPosition – function 1017H

Sent to request the Card ID currently playing a particular EGM position.

Additional Data:

<POSITION> - 32 bit (MSB first) EGM Position Number

### PlayerPreferenceReply – function 1018H

Sent in response to a RequestPlayerPreference message from the Maxgaming system.

Additional Data:

<CARD> - 32 bit (MSB first) Card ID

<Category> - 8 bit Category type ID as above

<Num> - 8 bit Number of preferences

Repeated for 1..Num

<Preference> - 30 BYTES – 8 bit ASCII text string describing the preferences e.g. “XXXX”

End Repeat

### TransactionValue – function 1019H

Optional transaction value in cents of any preceding transaction that day but preferably sent after a transaction success packet is received by the POS system.

Additional Data:

<TXN\_ID> - 32 bit (MSB first) Transaction ID.

<CENTSVALUE> - 32 bit (MSB first) Value in cents of transaction

### PurchaseItemList – function 1020H

The additional data for the message will be

<MSG\_ID> - 32 bit (MSB First) Message ID  
<INFO 0..N> - Ascii text in XML Format for Sales Information. See format below

This is the member sales information which will be sent from the POS to Maxgaming. The information will be sent shortly after the sale has completed at the POS. The ‘SalesId’ field will be the unique identifier for this sale for this venue. The “CardId” field will be non-blank if the sale is for a loyalty player.

Maxgaming should respond with a function to indicate successful receipt of the sale. The data in the reply should include the message id.

If no response is received within N seconds the POS will resend the message.

<Sale>

<SalesId>string-value</Saleid>

<TerminalId>string-value</TerminalId>

<TerminalName>stiring-value</TerminalName>

<LocationId>string-value</LocationId>

<LocationName>string-value</LocationName>

<CardId> long-value</CardId>

<SaleDate>“yyyy-mm-dd HH:MM:SS”</SaleDate>

<SaleTotal> int-value</SaleTotal>

<Items>

<Item>

<ItemCode>string-value</ItemCode>

<Description>string-value</Description>

<CategoryCode>string-value</Category>

<CategoryDescription>string-value</CategoryDescrption>

<UnitPrice> int-value</UnitPrice>

<CostPrice>int-value</CostPrice>

<Qty> int-value</Qty>

<SaleValue> int-value</SaleValue>

</Item>

*....*

</Items>

<Payments>

<Payment>

<PaymentCode>string-value</PaymentCode>

<Description>string-value</Description>

<IsPointsPayment>true-false-value</IsPointsPayment>

<IsCashPurchasePayment>true-false-value</IsCashPurchasePayment>

<TenderedAmount> int-value</TenderedAmount>

<ChangeAmount> int-value</ChangeAmount>

</Payment>

*....*

</Payments>

<Taxes>

<Tax>

<TaxCode>string-value</TaxCode>

<Description>string-value</Description>

<Amount>int-value</Amount>

</Tax>

*...*

</Taxes>

</Sale>

Refer appendix 5.3 for Maximum length and 5.4 for sample XML. Current Internal buffer for the request is 4KB excluding the XML nodes.

Refer appendix 5.5 to send Items/Payments/Taxes more than the Maximum limit.

# Appendices

## Maxgaming Card Format

The data encoded on the magnetic strip of the identification cards used in the Cougar monitoring system will be as specified in AS3524. The format of Track 1 data will conform to Track 1 Structure B of the standard as detailed below.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Description | Length | ASCII Rep. |
| STX | Start Sentinel | 1 Digit | % |
| FC | Format Code | 1 Digit | B |
| **PAN** | **Primary Account Number** |  |  |
| MII | Major Industry Identifier Number | 1 Digit | 9 |
| CC | Country Code | 3 Digits | 036 |
| II | Issuer Identifier Number | 4 Digits | 0027 |
| CRD | Unique Card Identification Number | 8 Digits |  |
| CD | Check Digit  Format: Luhn Formula - modulus10  As per AS3523 1988 | 1 Digit |  |
| **End of PAN** |  |  |  |
| FS | Field Separator | 1 Digit | ^ |
| NM | Name | 2 Chars | \_/ |
| FS | Field Separator | 1 Digit | ^ |
| ED | Expiry Date  Format: YYMM | 4 Digits |  |
| ID | Interchange Designator | 1 Digit | 7 |
| SC | Service Code | 2 Digits | 20 |
| DD | Discretionary Data | 9 Digits |  |
| ETX | End Sentinel | 1 Digit | ? |
| LRC | Longitudinal Redundancy Check  Format:  As per AS3522.2 1988 | 1 Digit |  |

## 16 bit CRC Generation

Seed Table and algorithm for 16 bit CRC.

/////////////////////////////////////////////////////////////////////////////

// 16-BIT CCITT CRC TABLE

static WORD g\_awCrc16Table[] =

{

0x0000, 0x1189, 0x2312, 0x329B, 0x4624, 0x57AD, 0x6536, 0x74BF,

0x8C48, 0x9DC1, 0xAF5A, 0xBED3, 0xCA6C, 0xDBE5, 0xE97E, 0xF8F7,

0x1081, 0x0108, 0x3393, 0x221A, 0x56A5, 0x472C, 0x75B7, 0x643E,

0x9CC9, 0x8D40, 0xBFDB, 0xAE52, 0xDAED, 0xCB64, 0xF9FF, 0xE876,

0x2102, 0x308B, 0x0210, 0x1399, 0x6726, 0x76AF, 0x4434, 0x55BD,

0xAD4A, 0xBCC3, 0x8E58, 0x9FD1, 0xEB6E, 0xFAE7, 0xC87C, 0xD9F5,

0x3183, 0x200A, 0x1291, 0x0318, 0x77A7, 0x662E, 0x54B5, 0x453C,

0xBDCB, 0xAC42, 0x9ED9, 0x8F50, 0xFBEF, 0xEA66, 0xD8FD, 0xC974,

0x4204, 0x538D, 0x6116, 0x709F, 0x0420, 0x15A9, 0x2732, 0x36BB,

0xCE4C, 0xDFC5, 0xED5E, 0xFCD7, 0x8868, 0x99E1, 0xAB7A, 0xBAF3,

0x5285, 0x430C, 0x7197, 0x601E, 0x14A1, 0x0528, 0x37B3, 0x263A,

0xDECD, 0xCF44, 0xFDDF, 0xEC56, 0x98E9, 0x8960, 0xBBFB, 0xAA72,

0x6306, 0x728F, 0x4014, 0x519D, 0x2522, 0x34AB, 0x0630, 0x17B9,

0xEF4E, 0xFEC7, 0xCC5C, 0xDDD5, 0xA96A, 0xB8E3, 0x8A78, 0x9BF1,

0x7387, 0x620E, 0x5095, 0x411C, 0x35A3, 0x242A, 0x16B1, 0x0738,

0xFFCF, 0xEE46, 0xDCDD, 0xCD54, 0xB9EB, 0xA862, 0x9AF9, 0x8B70,

0x8408, 0x9581, 0xA71A, 0xB693, 0xC22C, 0xD3A5, 0xE13E, 0xF0B7,

0x0840, 0x19C9, 0x2B52, 0x3ADB, 0x4E64, 0x5FED, 0x6D76, 0x7CFF,

0x9489, 0x8500, 0xB79B, 0xA612, 0xD2AD, 0xC324, 0xF1BF, 0xE036,

0x18C1, 0x0948, 0x3BD3, 0x2A5A, 0x5EE5, 0x4F6C, 0x7DF7, 0x6C7E,

0xA50A, 0xB483, 0x8618, 0x9791, 0xE32E, 0xF2A7, 0xC03C, 0xD1B5,

0x2942, 0x38CB, 0x0A50, 0x1BD9, 0x6F66, 0x7EEF, 0x4C74, 0x5DFD,

0xB58B, 0xA402, 0x9699, 0x8710, 0xF3AF, 0xE226, 0xD0BD, 0xC134,

0x39C3, 0x284A, 0x1AD1, 0x0B58, 0x7FE7, 0x6E6E, 0x5CF5, 0x4D7C,

0xC60C, 0xD785, 0xE51E, 0xF497, 0x8028, 0x91A1, 0xA33A, 0xB2B3,

0x4A44, 0x5BCD, 0x6956, 0x78DF, 0x0C60, 0x1DE9, 0x2F72, 0x3EFB,

0xD68D, 0xC704, 0xF59F, 0xE416, 0x90A9, 0x8120, 0xB3BB, 0xA232,

0x5AC5, 0x4B4C, 0x79D7, 0x685E, 0x1CE1, 0x0D68, 0x3FF3, 0x2E7A,

0xE70E, 0xF687, 0xC41C, 0xD595, 0xA12A, 0xB0A3, 0x8238, 0x93B1,

0x6B46, 0x7ACF, 0x4854, 0x59DD, 0x2D62, 0x3CEB, 0x0E70, 0x1FF9,

0xF78F, 0xE606, 0xD49D, 0xC514, 0xB1AB, 0xA022, 0x92B9, 0x8330,

0x7BC7, 0x6A4E, 0x58D5, 0x495C, 0x3DE3, 0x2C6A, 0x1EF1, 0x0F78

};

//@func Calculates a 16-bit CRC value given a buffer and a seed.

//@parm The buffer.

//@parm The size of the buffer.

//@parm The seed.

//@rdesc The final CRC value.

WORD JTCrc16( const BYTE \*lpBuff, DWORD dwLen, WORD wSeed )

{

assert( lpBuff != NULL );

// Calculate the CRC value.

WORD wCrc = wSeed;

for (DWORD n = 0; n < dwLen; n++)

{

// 1. x := <input\_byte> XOR <low CRC>

WORD x = (WORD) (lpBuff[n] ^ (WORD) (wCrc & 0x00ff));

// 2. x := CCITTCRC16TAB [x]

x = g\_awCrc16Table[x];

// 3. <low\_CRC> := <low\_fx> XOR <high CRC>

wCrc = (WORD) (x ^ (WORD) (wCrc >> 8));

}

return wCrc;

}

## Purchase Item Sale XML Node’s maximum length

<SalesId> string 15

<TerminalId> string 15

<TerminalName> string 20

<LocationId> string 15

<LocationName> string 20

<CardId> int

<SaleDate> date (yyyy-mm-dd HH:MM:SS )

<SaleTotal> int

<ItemCode> string 20

<Description> string 30

<CategoryCode> string 10

<CategoryDescription> string 30

<UnitPrice> int

<CostPrice> int

<Qty> int

<SaleValue> int

<PaymentCode> string 20

<Description> string 30

<IsPointsPayment> bool (0/1)

<IsCashPurchasePayment> bool (0/1)

<TenderedAmount> int

<ChangeAmount> int

<TaxCode> string 20

<Description> string 30

<Amount> int

## Purchase Item Sale XML content sample

<Sale><SalesId>SaleId101</SalesId><TerminalId>TerminalId101</TerminalId><TerminalName>TerminalName101</TerminalName><LocationId>LocationId101</LocationId><LocationName>LocationName101</LocationName><CardId>101</CardId><SaleDate>2012-06-29 11:06:49</SaleDate><SaleTotal>101</SaleTotal><Items><Item><ItemCode>ItemCode102</ItemCode><Description>Description102</Description><CategoryCode>CatCode102</CategoryCode><CategoryDescription>Category Description 102</CategoryDescription><UnitPrice>102</UnitPrice><CostPrice>102</CostPrice><Qty>2</Qty><SaleValue>102</SaleValue></Item><Item><ItemCode>ItemCode101</ItemCode><Description>Description101</Description><CategoryCode>CatCode101</CategoryCode><CategoryDescription>Category Description 101</CategoryDescription><UnitPrice>101</UnitPrice><CostPrice>101</CostPrice><Qty>1</Qty><SaleValue>101</SaleValue></Item></Items><Payments><Payment><PaymentCode>PaymentCode102</PaymentCode><Description>PaymentDescription102</Description><IsPointsPayment>0</IsPointsPayment><IsCashPurchasePayment>0</IsCashPurchasePayment><TenderedAmount>102</TenderedAmount><ChangeAmount>102</ChangeAmount></Payment><Payment><PaymentCode>PaymentCode101</PaymentCode><Description>PaymentDescription101</Description><IsPointsPayment>0</IsPointsPayment><IsCashPurchasePayment>0</IsCashPurchasePayment><TenderedAmount>101</TenderedAmount><ChangeAmount>101</ChangeAmount></Payment></Payments><Taxes><Tax><TaxCode>TaxCode102</TaxCode><Description>TaxDescription102</Description><Amount>102</Amount></Tax><Tax><TaxCode>TaxCode101</TaxCode><Description>TaxDescription101</Description><Amount>101</Amount></Tax></Taxes></Sale>

## Send Information more than the limit

For Sale request which have Items/Payments/Taxes greater than Maximum Buffer available one can send the information in batches with same SaleID

Ex: **Sale ID**: 1000

Items: 500 ; Item ID: 1..500

Payment: 100; Payment ID: 1..100

Taxes: 100 ; Tax ID: 1..100

In this scenario one can send XML request of batches of 10 items, 2 payments, 2 taxes with same Sale ID for an example.

The key is to have all the request to have same Sale ID so that all the Items/Payments/Taxes are updated under one Sale.

Batch 1 Sale ID 1000 and other nodes with item ID 1..10, payment ID 1..2, tax ID 1..2

Batch 2 Sale ID 1000 and other nodes with item ID 11..20, payment ID 2..4, tax ID 3..4

and so on.

One may be able to send more items/payments/taxes as it is dependent on the Size of content of each element. Each request should be a valid XML Sale request i.e it should not miss any elements as defined in 6.4.15