

Cash Register - Cortex integration

Integrating Carwash POS/loyalty system with Kesseltronics carwash management backend.

Author : Jaap Versteegh <j.r.versteegh@orca-st.com>
Created : 2010-10-19
Edited : 2013-05-17
Revision : 35

Contents

1 Introduction.....	3
2 General structure.....	4
2.1 XML-RPC API.....	4
2.2 GetVersion.....	4
3 Transaction information for management database.....	5
3.1 Execute.....	5
3.2 Annul.....	6
3.3 Refund.....	6
3.4 Query.....	6
4 Authentication/authorization.....	7
4.1 Authenticate.....	7
4.2 AuthorizePayment.....	7
4.3 AnnulPayment.....	7
Abbreviations.....	8
References.....	8

1 Introduction

ACE Carwash Systems B.V. wishes to offer its customers a complete carwash software solution. It intends to do so using Cash Register/ POS software and Kesseltronics "Cortex" management software.

This document contains a plan/design for integration of the Carwash POS system with the Cortex carwash management system. In order to provide a customer with a single management interface, transactional information needs to be transferred from Cash Register to Cortex.

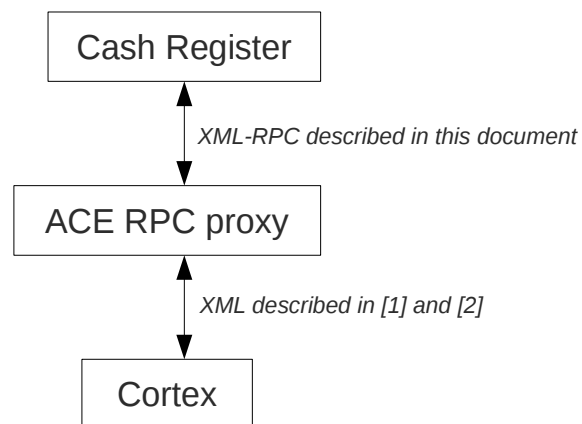
The second chapter discusses another desired feature, which is an external authentication and authorization option for a loyalty system. This allows carwash owners to join systems like the "Nationale Waspas", an external database with value cards/codes.

2 General structure

The proposed communication method for the exchange of desired information is XML-RPC [4]. This is a simple, well specified protocol for sending messages using XML over an HTTP connection. XML-RPC is also used for communication with ProgramPasser [3], so using it will help create a more uniform interface for all carwash related RPC.

The communication interface to Cortex [1] and Webcode [2] is based on sending custom XML messages over an HTTP connection, so a proxy service will need to be provided by ACE, to translate the XML-RPC calls to Cortex XML. This service can run locally on the cash register computer, a computer central to the washcenter or the cortex server.

The RPC proxy will return a version number when requested, so a client can determine its capabilities. The next two chapters describe the current implementation.



2.1 XML-RPC API

The API is provided here using pseudo code function prototypes. All types are XML-RPC standard types and square brackets denote an array of the specified type.

2.2 GetVersion

```
int CasAPI.GetVersion();
```

Returns the API version provided by XML-RPC service. Current version is 3.

3 Transaction information for management database

Three messages are provided to notify the management database of transactions at the cash register.

- *Execute*
Notify of a transaction at the cash register
- *Annul*
Annul (part of) an earlier transaction. e.g. to correct cashier error
- *Refund*
Refund the customer for product. e.g. because of customer dissatisfaction.
- *Query*
Get info on previously executed transactions

3.1 Execute

```
struct Product {
    int Code;           // Product identifier
    int PriceEx;        // Full product price excluding VAT (cents)
    int PriceInc;        // Full product price including VAT (cents)
    int Discount;       // Promotion value (cents)
    string DiscountType; // Promotion type identifier
    string PaymentType; // 'money' | 'credit' | 'voucher'
    [int BalanceCharge;] // When the product is associated with
                        // a change of customer balance, this
                        // is the amount charged in cents
};

struct Voucher {
    int Code; // Unique code of voucher
    int Value; // Value of voucher in cents
    string Type; // Type of voucher
};

struct Payment {
    int Amount; // Amount (cents)
    string Type; // 'cash' | 'pin' | 'chip' | 'creditcard' |
                // 'sms' | 'online' | 'eft' | 'ideal' | 'paypal'
};

boolean Transaction.Execute(
    dateTime.iso8601 TransactionTime;
    int CashRegisterId; // Unique identifier for cash register
    int CashierId;      // Unique identifier for cashier
    Product[] Products;
    [Payment[] Payments;] // Only required when a payment is made
    [Voucher[] Vouchers;] // Only required when voucher redeemed
);
```

3.2 Annul

```
boolean Transaction.Annul(  
    dateTime.iso8601 TransactionTime; // Time of original  
                                     transaction.  
    dateTime.iso8601 AnnulTime; // Time of annulment  
    int CashRegisterId; // Unique identifier for cash register  
    int CashierId; // Unique identifier for cashier  
    Product[] Products;  
    [Voucher[] Vouchers;] // Only required when voucher returned to  
                           customer.  
                           // Return payment is assumed to be in cash  
);
```

3.3 Refund

```
boolean Transaction.Refund(  
    dateTime.iso8601 RefundTime; // Time of refund  
    int CashRegisterId; // Unique identifier for cash register  
    int CashierId; // Unique identifier for cashier  
    Product[] Products;  
                           // Return payment is assumed to be in cash  
);
```

3.4 Query

(added in v3)

Query the management database for results

```
struct QueryResult {  
    string Id; // For Type "payment":  
               //'unknown'|'cash'|'eft'|'online'|'ideal'|  
               //'paypal'|'chip'|'creditcard'|'sms'|'account'|  
               //'prepay'|'credit'  
    // For other Type:  
    // ID of product/product group or voucher  
    int Amount; // For Type "payment" and "productvalue":  
                // amount in cents  
                // for Type: "product" and "voucher": a quantity  
};  
  
struct QueryResults {  
    string Type; // If all is well: copy of Type queried  
                // otherwise: 'none'  
    QueryResult[] Results;  
};
```

```
QueryResults Transaction.Query(  
    dateTime.iso8601 StartTime;    // From this datetime  
    dateTime.iso8601 EndTime;      // ... to this datetime  
    string Type;                   // Type to query: 'payment' | 'product' |  
                                   'voucher' | 'productvalue'  
    [int CashRegisterId;]          // -1 means all  
    [int CashierId;]               // -1 means all  
    [string Id;]                   // Id of "Type" to query. All are returned when  
                                   omitted or empty string  
);
```

4 Authentication/authorization

External authentication and authorization should allow a customer with an (RF)ID card that is not known to the cash register, to pay for wash programs. The interfaces allow the retrieval of data associated with an ID and authenticate a financial transaction.

4.1 Authenticate

```
struct IdRecord {
    boolean Valid; // True when the Id exists
    boolean Active; // Id is active
    int Balance; // Amount spendable in cents
    string Name; // Optional name associated with Id
    string Message; // Optional message associated with Id
    [dateTime.iso8601 LastVisit;]
                        // Last time this Id was used (added in v2)
};

IdRecord Identification.Authenticate(
    string Id; // Id of coupon/customer
);
```

4.2 AuthorizePayment

```
struct AuthRecord {
    boolean Success; // Whether authorization was successful
    string Reason; // Message (in case of failure)
    [string PaymentId] // Payment Id (added in v2)
};

AuthRecord Identification.AuthorizePayment(
    dateTime.iso8601 TransactionTime;
    int CarwashId; // Unique identifier for car-wash company
    int CashRegisterId; // Unique identifier for cash register
    int CashierId; // Unique identifier for cashier
    string Id; // Id of customer/coupon
    int Payment; // Amount in cents
);
```

Return a record with result of payment attempt.

4.3 AnnulPayment

(added in v2)

```
AuthRecord Identification.AnnulPayment(
    string PaymentId; // Payment Id of previous payment
);
```

Returns an AuthRecord with annulation result (True when the payment was annulled. False otherwise) and a reason in case the annulment failed.

Abbreviations

API	Application Programming Interface
HTTP	Hypertext Transfer Protocol (http://www.w3.org/Protocols/rfc2616/rfc2616.html)
XML	eXtensible Markup Language
RPC	Remote Procedure Call
XML-RPC	eXtensible Markup Language Remote Procedure Calling (http://www.xmlrpc.com)

References

- | | | |
|---|----------------|---|
| 1 | KesselTronics1 | "cortex Interface Control Document 091410.docx" |
| 2 | KesselTronics2 | "cortexCode Interface Control Document.docx" |
| 3 | J.R. Versteegh | "ACE ProgramPasser RPC" |
| 4 | XML-RPC | http://www.xml-rpc.com/ |