CITY OF CAPE TOWN
ISIXEKO SASEKAPA
STAD KAAPSTAD

ERP Support Centre

CCT Customer Service Requests

Application Programmers' Interface

Last update:	19 August 2024
Owner:	Gareth Döhne
Version:	01
Status:	Draft version 03

Document Control Information

Effective From	Amendment	Nature of Change	Amended By	Reviewed By	Approved By
2020-06-23	Document creation	Major	Gareth Döhne		
2016-04-04	Rewrite of user registration	Major	Mikael von Ketelhodt	Gareth Döhne	Gareth Döhne
2018-01-22	Added file attachment parameters to 4.6.1	Minor	Gareth Döhne		
2018-09-11	Cosmetic changes and typographical corrections	Minor	Gareth Döhne		
2020-06-24	Minor restructuring of content to clarify use	Minor	Gareth Döhne		
2024-08-17	Terms & Conditions update	Major	Cassiem Mapker	Helourine Seyffert	Melanie Sonnenberg

Helourine Seyffert
Digitally signed by Helourine Seyffert
Date: 2024.08.19 18:11:03 +02'00'

Table of Contents

List of Figures

List of Tables

GLOSSARY

Term	Description
Account Number	Account Number refers to the number of the municipal account statement that the City delivers to the property owner (or his/her nominee) on a monthly basis. The Account Number is prominently recorded on the account statement at the top right of the page.
Account-Based Service	An Account-Based Service is a Service for which the Service Location is determined using an Account Number. For example, a request to carry out a physical meter reading is an Account-Based Service. The Account Number positively identifies the specific property at which an Account-Based Service must be delivered and is the most accurate mechanism for defining the Service Location. Account-Based Services are typically requested by the owner of the property concerned, who should have access to the Account Number for that property through the receipt of his or her municipal account statement.
API Consumer	An API Consumer (or just Consumer), in this document, refers to the Third-Party that is producing or has produced a software application using the application programmers' interface described in this document (i.e. a Consumer of the API).
C3 Notification	A specific type of Service Request that the City has been using for Customer requests and complaints. The generic term "Service Request" is now preferred as the City is moving away from the use of the C3 notification type for customer complaints and requests.
Categorisation	The process of selecting a specific Service, in this case using a two-level hierarchy of a grouping of related Services at the first level and the specific Service at the second.
Contact Details	Contact Details include the name, telephone number and possibly email address of the person requesting a specific Service. Contact Details are important in the event feedback is required for the Service Request but, more importantly, in the event there is insufficient or ambiguous information in the definition of the Service Request, in which case this detail is used by the City to contact the person for additional information.
Customer	A Customer, as used in this document, refers to a member of the general public that wishes to lay a complaint with, or request a service from the City. This includes residents, visitors and any entity that wishes to engage with the City through the Service Catalogue offered through this interface.
General Service	A General Services is one where the Service Location cannot readily be identified by an Account Number. For example, a pothole or burst water main is a General Service. The Service Location for a General Service is typically identified by pointing out the location on a map.
Service	A Service (also known as Subtype) is the specific activity or set of activities that a Customer could request the City to deliver. A Service is selected from a Group of similar Services (also known as the Service Catalogue). The grouping of services is used merely to facilitate selection as the list of Services available through this interface are too numerous to make it practical to select individual Services from a single, flat list, hence the need to group them into a hierarchy.
Service Catalogue	The Service Catalogue is the full set of Groups (Types) and Services (Subtypes) that are made available to Consumers and Customers through the application programmers' interface described here. Consumers may choose to create their own Service Catalogues that would necessarily have to be a sub-set of the City's Service Catalogue and that would have to map directly to the City's catalogue.
Service Class	Service Class in this document refers to the classification of Services either as Account-Based or as General Services.
Service Group	A Service Group (or Group, also known as Type) refers to a logical grouping of Services into a hierarchy to facilitate the selection of a specific Service. The City offers too many individual Services through this interface to make it practical to select individual Services from a single, flat list, hence the need to group them into a hierarchy.
Service Location	The Service Location refers to the specific location at which a requested Service is to be delivered. It is imperative that the location be properly identified so that the City can determine where to deliver the requested service. Service Locations for Account-Based Services are determined by the Account Number for the property concerned. Service Locations for General Services are typically identified either by a latitude and longitude GPS coordinate pair or by a street address. Identifying the Service Location on a map is usually the surest way to capture this information.

Term	Description
Service Request	The generic term for the electronic document (or business object) that is used to capture a request to deliver a specific Service, to assign the piece of work to a specific service department, to execute and record the delivery of that Service and to measure and report on the Service and the delivery of that Service by the service department.
Session	A Session in the context of this document refers to a logically related set of activities that a user of the consumer's application would execute in one interaction, typically culminating in the creation of a Service Request or the closing of the application by the user. Sessions are used in the City's systems to facilitate the tracing of a user's activities in the event of an error or unexpected result.
Subtype	A Subtype, also known as a Service, is a specific Service that a customer could request the City to deliver. Services are hierarchically organised into Types (or Groups) and Subtypes (Services). The terms Type and Subtype are used to describe these elements on a technical level while the terms Group and Service are used to describe these at a user-interface level.
Third-Party	A Third-Party, also known as a Consumer, in the context of this document refers to the entity that consumes the application programmers' interface described in this document.
Type	A Type, also known as a Group, is a logical, grouping of related Subtypes (or Services) into a hierarchical structure to facilitate selection of specific Services (Subtypes). The terms Type and Subtype are used to describe these elements on a technical level while the terms Group and Service are used to describe these at a user-interface level.
SSO2 Cookie	An authentication token that is returned as a result of user login. Including this cookie in relevant endpoint operations provides user context to the service.

1 Executive Summary

This document describes an application programmers' interface, developed by the City of Cape Town, for the maintenance of customer service requests. This interface makes it possible for registered third parties to develop their own, typically community service, applications that include the capability to register complaints or requests for service directly with the City.

This interface integrates with what was previously referred to as the City's "C3 Notification" process, which is the mechanism the City uses to capture requests from the general public for specific tasks to be completed, be it the repair of a pothole or the removal of a whale carcass from the shoreline.

The primary purpose in building this interface was for the City to deploy its own customer-facing, self-service web applications in a structured manner with the additional objective of engaging in partnerships with community-focussed organisations wanting to bring these kinds of services to their user communities. This kind of collaboration is aimed at developing a safer and better run City ultimately aimed at providing improved opportunities for all of its residents and visitors.

This document describes both the technical implementation of the application programmers' interface as well as the registration process for organisations wanting to make use of this interface.

2 API Description

This section provides an overview of all aspects of the application programmers' interface, including registration to use the interface, the various resources that constitute the interface and how these should be orchestrated into an application that will deliver consistent and reliable results.

There are a number of aspects to the use of the interface that are not complex but should be clearly understood at the outset. These include:

- Registration to use the API
- Security mechanisms
- REST Resources and how these should be orchestrated, covering,
    - Session Management
    - Domain & Configuration data
    - Account data
    - Service Requests and
    - Users

This section provides a descriptive overview of the concepts and their implementation and of the resources and their use.

In terms of the logical progression of activities, any organisation wanting to make use of the API must request permission to do so from the City in a registration process. There are no registration fees or fees for the use of the API but the Consumer must provide a simple undertaking regarding the use of the interface and the taking of necessary precautions to safeguard the City's environment.

2.1 Introduction

The interface described here is implemented using RESTful web services designed following the guidelines in the article at http://msdn.microsoft.com/en-us/library/dd203052.aspx. (The implementation does however not make use of the Windows Communication Framework described in the above article. It is a SAP ABAP NetWeaver solution.)

Message payload is delivered using the lightweight JavaScript Object Notation (JSON).

Security (and some authentication) is ensured through the use of public and private keys issued to registered consumers of the interface and the use of keyed-hash message authentication codes (HMAC) for key services (similar to the implementation used by Amazon Web Services).

Data exchange with the City must use HTTPS via SSL. This is the only supported protocol for external access.

The City exposes a quality assurance system for Consumers to use both for development and testing and a live system for productive operations. Once Final Acceptance Testing is concluded, the Consumer's application can be published against the City's live (production) system. The systems are described further in section 5, Quality Assurance & Production System Hostnames.

2.2 Authentication & User Management

The deployment of the API will make use of a guest account to login to the City's servers. Guest account authentication is performed when the guest login service is called. The guest account credentials are hosted on the server being called so the consumer does not need to provide any credential information in order to be able to access the service.

In order to login, make use of a GET request to the /api/zcur-guest/login endpoint services to login and retrieve an SSO2 authentication cookie. Once a cookie has been received, you can include that into the create service request HTTP requests.

A service is also provided for retrieving the currently logged in user (which requires an SSO2 cookie). This service is used primarily to check the validity of the SSO2 cookie. If it has expired, the user should be prompted to login again to obtain a new cookie, failing which the user context will be lost and the communication will revert to a guest account.

A second level of authentication is implemented through the use of key-hashed message authentication codes (HMAC). (See section 2.3, Message Authentication (HMAC)and the header parameter X-Signature on page 20.)

2.3 Message Authentication (HMAC)

Message authentication is performed on all HTTP POST operations. This will most typically be the service request create service.

Authentication is carried out using base 64 encoded, keyed-hash message authentication codes (HMAC). The hash code is calculated using the SHA256 algorithm and encrypted using the private key supplied to the consumer at registration. The JSON string forming the message body is the value that is hashed. The resulting code is base 64 encoded.

The consumer application forms the message body string and then calculates the HMAC SHA256 authentication code using this string and the private key. The code is then base 64 encoded. This string is inserted into the relevant message as an HTTP header parameter named X-Signature.

The City's services calculate the HMAC independently using the body from the HTTP message and the private key that the City has on record for the consumer, determined from the associated public key that is provided in the HTTP header parameter X-Service. The City's base 64 encoded HMAC is then compared for equality with the HMAC supplied by the consumer. The POST operation is carried out if the HMACs are equal. An exception is raised if there is a difference between the (encoded) HMAC values independently computed.

Guidelines are available in the Internet for the creation of hashes as described above. One such resource (with support for multiple programming languages) can be found at http://www.jokecamp.com/blog/examples-of-creating-base64-hashes-using-hmac-sha256-in-different-languages/.

2.4 HTTP Header Parameters

The header parameters used in this API are defined in Table 1: HTTP Header Parameters in section 4.1 (page 21). Header parameters are mandatory, depending on the service, and are used in this API:

1) To identify the consumer application; parameter X-Service (page 20) is used to transport the public key issued to the consumer.
2) For session management; the parameter X-Session (page 20) is used to transport the session identifier, which is obtained from the "session" resource (see also section 2.5 below).
3) For supplying keyed hash codes in the case of post operations as a security measure; the parameter X-Signature (page 20) is used to transport the HMAC value.
4) In order to (optionally) authenticate a registered user against the service, the sap-alias and sap-password headers need to be populated when POSTing to the /api/zcur/users service (see section **Error! Reference source not found.**, **Error! Reference source not found.**, on page **Error! Bookmark not defined.**).

2.5 Sessions

The concept of a session has been introduced to assist in the process of logging and tracing. A session, in this context, is intended to be a series of interactions between an individual user and the City in one "session". A session is identified by a session key, which takes the form of a 32-character globally unique identifier (GUID).

The City records a history of interactions between the user and the API services in a log of activities. This log is used to assist in tracing issues in the event of an error or unexpected result. The activities are grouped into sessions so all activities for a single session are arranged as one log entry with multiple messages in that log (one for each activity).

It is possible to have a single log entry for all activities in all sessions for all users of a particular consumer application but that would defeat the purpose of the grouping and would eventually create a performance bottleneck. Session management is up to the consumer application but the recommended guideline is as follows:

- Request a new session identifier prior to the first operation for every user when they initiate the consumer application component that interacts with the City's API.
- Use that same session identifier for all of the interactions of that user until such time as a new service request is created.
- Request a new session identifier after service request creation and before any subsequent operation.
- Clear the session identifier after the user leaves the component so that a new identifier is issued on any subsequent access.

The session service is defined in section 4.3 Resource: Session (on page 22) of this document. (See also Table 5: API-WSR-SE01 – Get Session Identifier).

2.6 Domain & Configuration Data

Domain and Configuration data are used during the creation of a service request to specify the specific service for which the request is to be created and to determine the service class (account-based or general), which informs the data that needs to be captured for the service request.

This data is crucial to the successful integration and must be incorporated into the consumer's solution.

There are two domains, together forming the service catalogue:

- The list of type codes and descriptions and, (for each of the types),
- The list of subtype codes and descriptions.

There is one set of configuration data that is used to determine the service class for each subtype. The service class determines whether the application must require an account number (for account-based services) to determine the service location or whether the service location may be defined by GPS coordinates or a street address.

2.6.1 Online Integration

The domain and configuration data can be consumed interactively using the City's API services. For interactive consumption there are services that deliver:

- The current list of type codes and descriptions.
- For a specific type code, the current list of subtype codes and descriptions.

2.6.2 *Staging Domain & Configuration Data*

The domain and configuration data are very static. The consumer may therefore elect to stage this data on their server or client application. If this route is chosen some mechanism should be put in place for periodic refresh or review of the data to pick up any occasional changes that may be made by the City. For staged data there are services that deliver:

- A complete list of type codes and descriptions.
- A complete list of type codes, subtype codes and descriptions of subtypes.
- A complete set of service class configuration for all type and subtype codes.

2.6.3 *Custom Service Catalogues*

Consumers may also elect to define their own service catalogue that would necessarily have to be a subset of the City's catalogue. If this option is chosen, the consumer will have to put in place a mapping service to convert the consumer's service catalogue entries into the City's type and subtype codes (and vice versa).

2.7 Account Data

The Account resource delivers information about an account and is relevant for account-based services only. It is not essential that this be deployed but that depends to a degree on how the configuration data is served. There are two operations in this space, described below.

2.7.1 *Service Class*

The first service provides a true or false response to whether or not a specific service (type and subtype combination) is an account-based service or not. This returns the same information that the configuration data service described under section 2.6.2 returns, however that service returns this detail for all services. This service returns the class indicator for a specific service and is required only if the configuration data is not staged and the consumer application delivers both account-based and general services.

2.7.2 *Account Details*

This service returns selected details about an account, most important whether the account number is valid or not and, for valid accounts, the location of the property, which is pertinent if the application shows the location of the service on a map.

Account-based services will only be created if a valid account number is submitted so this validation is critical to the consumer's application if the application is going to expose account-based services.

Accounts and account details are dynamic and change on a daily basis. That, coupled with the sheer volume of active accounts in the City, makes the staging of this data impractical with the current services. This validation is consequently only available interactively.

2.8 Service Requests

2.8.1 *Service Request Creation*

Description

The primary purpose of this interface is to provide the general public with a self-service mechanism to request the City to perform specific services, for example to repair a pothole, check a faulty meter, and trim trees that are becoming problematic and etcetera.

Requests of this nature that are submitted to the City culminate in a service request, which is an electronic document in the City's IT systems in which the requested service is recorded together with a location identifying where that service is to be performed as well as the details of the person requesting

the service. The customer's details are important for the purpose of clarifying details should this not be clear from the information in the service request as well as to provide feedback where appropriate.

The service request (electronic document) is routed directly to the department responsible for delivering the requested service based on the type of service and (possibly) the location of the requested service.

Every electronic document representing a service request has a unique number, commonly referred to as the reference number. This reference number can be used on any channel of interaction with the City (self-service, call centres and walk-in centres) to follow-up on the status of the requested service.

The services available via this self-service channel are a subset of the full, service catalogue supported by the City. The services that are not available via this channel have been excluded for specific reasons, usually related to process execution and the efficiency of that process. These (excluded) services are available via other customer-service channels such as call centres, walk-in services and subscribed notification services.

The City has public facing applications that make use of the interface described here. This interface is now also available to third parties wanting to develop their own application components that integrate directly into the City's customer service framework.

Service Request Create Process Flow

Figure 1 below shows the flow of application steps necessary for the creation of a service request. (See also section A.2, Service Request Create, for an example of the screen flow for creating service requests.)

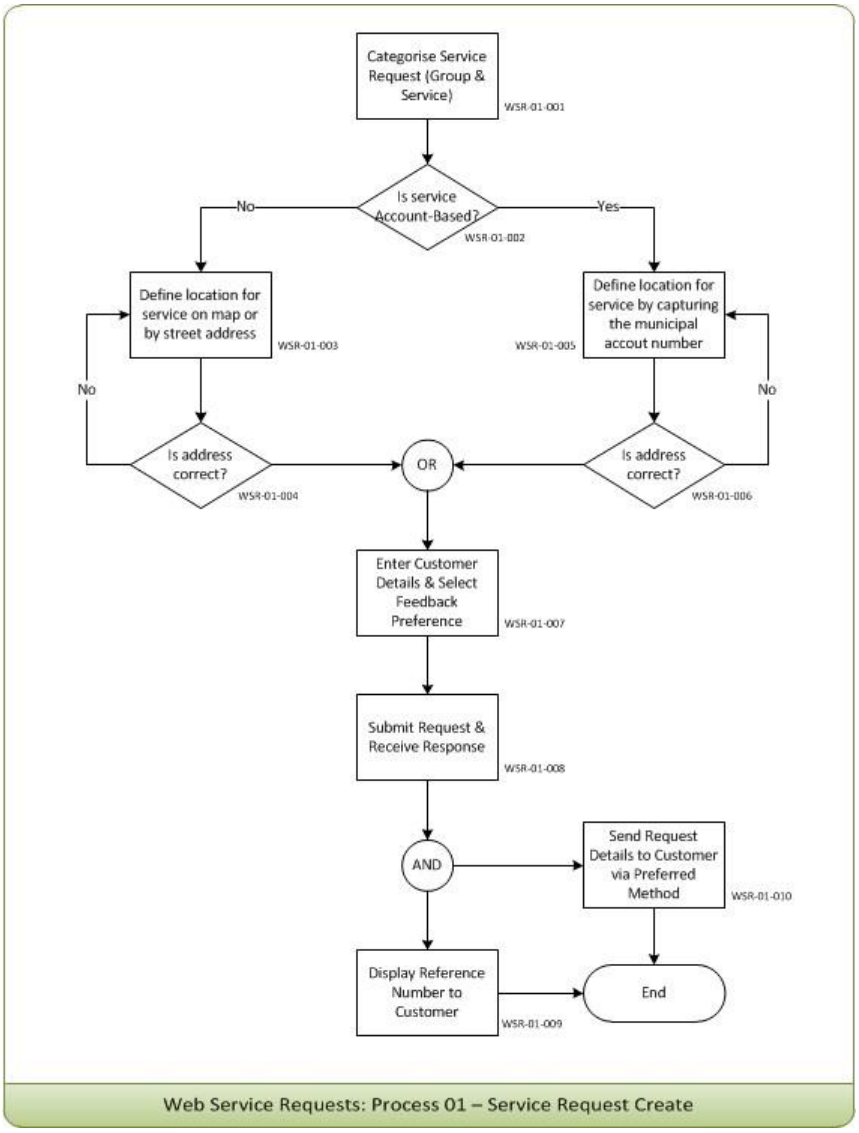

Figure 1: Service Request Create Flow Logic

Process Description

Step	Description	Comments	Resource relative /coct/api/zsreq
WSR-01-001	Categorise Service Request (Group & Service)	Categorisation consists of selecting a group (type) code and a service (subtype) code from a predefined set. The set of group definitions are delivered as a pair of codes and associated descriptions (type, name), sorted by description.	/config/types
		Once a group is selected, the user is required to select a specific service in that group. The set of services associated with a specific group are delivered as a triplet of group code (type), service code (subtype) and description (name), sorted by description.	/config/types/{typeid}

Step	Description	Comments	Resource relative /coct/api/zsreq
		Group and service codes and descriptions are reasonably static. This data can be staged by the third party and refreshed periodically as deemed necessary. An additional service is provided for this purpose that delivers all of the services (subtypes) for all groups (types).	/config/subtypes
WSR-01-002	Is the selected service a general or an account-based service?	Services are broadly grouped into two classes, general services and account-based services. In this step, the process determines into which class the selected service falls in order to decide which approach to adopt for determining the location for the service delivery. For general services proceed to step WSR-01-003. For account-based services, proceed to step WSR-01-005.	/account/ required?type={typeid}& subtype={subtypeid}
WSR-01-003	Define location for service on map or by street address	General services are those delivered to members of the public where the location of the service is independent of the property owner. The location of the service is defined by a street address or a latitude / longitude location obtained from a device or located on a map. The City currently uses a Google Map API for this purpose but intends to move to a City map viewer in the foreseeable future.	None (External to this API)
WSR-01-004	Confirm service delivery location	The street address determined from the spatial location or street address directly is displayed for confirmation by the application user. If the address is rejected, execution returns to step WSR-01-003. If the address is confirmed execution continues with step WSR-01-007.	None (User confirmation dialogue)
WSR-01-005	Define location for service by capturing the municipal account number	Account-based services are those delivered for the benefit of an owner of a specific property. The location of the service is defined by the location of the property, which is unambiguously identified by supplying the municipal account number for that property. The account number provided by the user must be validated against the City's database to ensure it is a valid account number. The validation service (Get Account Details) returns the street address of the property in question.	/account/{accountno}/validate
WSR-01-006	Confirm service delivery location	The street address determined from the municipal account number is displayed for confirmation by the application user. If the address is rejected, execution returns to step WSR-01-005. If the address is confirmed execution continues with step WSR-01-007.	None (User confirmation dialogue)
WSR-01-007	Enter Customer Details & Select Feedback Preference	The name and contact details of the requestor are captured at this point, together with their preference of channel for feedback.	None (User dialogue)
		A mobile number is required, the form of which is required to be exactly 10 numeric digits (after all punctuation and whitespace is removed).	
		If an email address is supplied, it must conform to the standards for email address definition.	
		If the preference is for email feedback, the email address is a mandatory field.	

Step	Description	Comments	Resource relative /coct/api/zsreq
WSR-01-008	Submit Request & Receive Response	The information collected in the steps above is assembled into and POSTed to the City's service, together with an HMAC code. The response will be an error message in the event of failure or, in the event of a successful post, the service request reference number together with a summary of the request details	/sr
WSR-01-009	Display reference number to customer	The consumer's application should deliver, at a minimum, the number of the created service request for reference.	None (User dialogue)
WSR-01-010	Send Request Details to Customer via Preferred Method	If the preferred feedback channel was selected and the request was created successfully, a confirmation of the request reference number and detail will be sent directly to the end user on the preferred channel.	None (System response)

2.8.2 Service Request Status

The interface includes a service to query the status of an existing service request, given the service request number. Registered users can also be provided with an historical list of the service requests they have opened but that is limited to users that have registered or that have been registered on the City's database.

When querying the status of an existing service request, the system returns a string containing a description of the current state of the request, which can sometimes be confusing but which does add value in most instances, together with some other details including the current long description details.

The long description is often updated by the City's staff to reflect progress, queries or other activities that have occurred in relation to the specific service request. This is however not always done and when done is not done consistently so there is not always value to be had from this data attribute.

2.9 User Registration & Administration

User registration and the services available to registered users are all optional and not critical to the process of creating service requests and following up on the status of a service request. There is some value to be added for registered users such as pre-filling fields from the user's profile instead of having to re-enter the data every time and in keeping a handy, historical record of service requests that have been created by a specific user.

NOTE: THE USER REGISTRATION AND ADMINISTRATION OPERATIONS ARE NOT YET COMPLETE OR FULLY DESCRIBED IN THIS DOCUMENT. FURTHER DEVELOPMENT OF THESE SERVICES IS CURRENTLY ON HOLD. CONSUMERS WILL BE ADVISED OF ANY CHANGES IN THIS STATUS.

2.9.1 User Registration (Deprecated)

The User resource is used to register and manage users in the City's environment. The use of this resource is deprecated. Consumers are advised to avoid implementation of the user registration services until further notice.

2.9.2 User Login

Registered users can login with their username and password. This would typically be used to create a cookie containing the user's default details that can then be used to pre-populate the forms used to capture new service requests.

Consumers of the API that have not been provided with credentials to use this service are advised to make use of the guest account login (see section 4.2.1, API-WSR-US22: Login using a Guest User account).

2.9.3 Profile Maintenance (Deprecated)

A profile maintenance utility should be provided for registered users that will enable the user to change details, add additional mobile device numbers to their profile and maintain those recorded against their profile, add account numbers to their profile and maintain those recorded against their profile and reset their passwords.

2.9.4 Service Request History (Deprecated)

A service is provided that can be sued to deliver a history of the service requests a user created when logged in for reference purposes.

3 Third-Party API Consumer Registration

The API consumer registration process, for now, will be manual via email interaction with the City. Terms and conditions of usage of this API are also still to be defined. The terms and conditions should be in line with the below principles and not limited too:

- Re-logging of the same request needs to be avoided as it leads to inappropriate use of the system and should rather indicate that the related request has been logged. They should allow the user to receive their own reference number so that they can receive a status update if required from the user, otherwise just have a notice displayed to indicate that the call was logged before.
- For all City Service Requests logged via the app, the City generated service request number **must** be provided to customers. References that gives the impression that external app transactions comes from the City needs to be refrained from, e.g. the use of "CCT" as part of the number.
- Service Request number should only be provided to the customer once the transaction is logged on the City's side.
- State that companies that are using the API needs to update their 1) FAQs and need to provide 2) pop-ups that states that the company will comply with POPIA for all personal information and other data collected via their application. That the City is not responsible for any data outside of the City's domain. Further state that the City's relationship is with the company and not directly with the citizen.

Essentially the use of this API is intended to be a lightweight engagement with consumers doing due diligence to ensure the City's resources are not abused, to ensure that they do not create test data in the City's live environment, leading to wasted effort and resources and to have a common understanding that the City reserves the right to withdraw a consumer's access to the API in the event that abuse or inappropriate use is detected.

The City's objective in making this API available is to form mutually beneficial partnerships with entities that have a community-based focus and wish to include the City in their service offering.

Registered consumers will be issued with a public and private key that is used to authenticate message data in specific cases. The private key is intended to be kept private and the public key is shared in every message. (See section 2.3 Message Authentication (HMAC) for further details.)

4 REST API Resource Description

This part of the document describes the API resources in some detail. Web description language versions of the API have not been developed at this point. There is no intention to do so in the near future however the City is open to suggestion on how it can improve the API and the documentation of the API.

NOTE: THE RESPONSES FOR THE INDIVIDUAL OPERATIONS FOR EACH RESOURCE HAVE NOT YET BEEN DOCUMENTED HERE. THESE WILL FOLLOW IN THE FINAL RELEASE OF THIS DOCUMENT.

NOTE: THE DESCRIPTIONS OF THE OPERATIONS FOR THE SESSION, CONFIGURATION, ACCOUNT AND SERVICE REQUEST RESOURCES ARE MORE-OR-LESS COMPLETE WITH THE EXCEPTION OF THE RESPONSE VALUES.

THE DESCRIPTIONS OF THE USER RESOURCE OPERATIONS ARE MORE-OR-LESS COMPLETE UP TO SECTION ***ERROR! REFERENCE SOURCE NOT FOUND.****. SECTIONS REF _REF405304192 \R \H* ***ERROR! REFERENCE SOURCE NOT FOUND.*** *THROUGH TO* ***ERROR! REFERENCE SOURCE NOT FOUND.*** *ARE STILL TO BE UPDATED.*

4.1 HTTP Header Parameters

The API has mandatory header parameters that must be provided with every call. Table 1: HTTP Header Parameters lists and describes the HTTP header parameters in use in this interface.

Parameter Name	Description	Value characteristics
X-Service	This is the value of the public key issued to the registered consumer by the City. The key: • Must be provided as a mandatory parameter in all messages, • It must exist in the City's database of registered consumers and • It must be an active key. (Keys may be switched to inactive when a consumer no longer wishes to interact with the City or in the event of malicious attacks originating with the issued key.) If any of the three conditions above are violated, an HTTP 500 response is issued with an appropriate short text error message. (See also section 3, Third-Party API Consumer Registration.)	GUID of 32 characters with no punctuation. For example: F6E9F4F4833B1ED493D848A953009F35
X-Session	This is the session key obtained periodically from the City representing the log file entry of a user's activities (as recommended in section 2.5 Sessions). The session identifier must be provided in all calls other than the service that returns the session identifier. If the session identifier is missing from a call, an HTTP 500 response is issued with an appropriate short text error message.	GUID of 32 characters with no punctuation. For example: F6E9F4F4A40C1ED49AAE6F2ECAE515E5
X-Signature	This is a hexadecimal string containing a key-hashed message authentication code (HMAC) calculated from the message string using the SHA256 hash algorithm and encoded using base 64. The Signature parameter: • Is mandatory for POST operations, • Must contain a correctly encrypted code based on the message content, • Uses the private key issued by the City to the consumer for encryption. If any of the above is violated, an HTTP 500 response is issued with an appropriate short text error message. (See also section 2.3, Message Authentication (HMAC).)	Base 64 encoded string containing an HMAC SHA256 authentication code.
sap-alias	This is the username of the registered user on the SAP system. This should be included in POST /api/zcur/users operation in order to get a valid SSO2 token (on valid authentication to the service).	A string not greater than 12 characters long.

Parameter Name	Description	Value characteristics
sap-password	This is the password of the registered person on the SAP system. This should be included in POST /api/zcur/users operation in order to get a valid SSO2 token (on valid authentication to the service).	A string value.

Table 1: HTTP Header Parameters

Not all resources require the same parameters but, briefly:

- The X-Service parameter is mandatory on all requests.
- The X-Session parameter is mandatory on all requests with the exception of the Get Session Identifier service (see section 4.3.1, Session Operation API-WSR-SE01: Get Session Identifier) that is used to get a value for the X-Session parameter and the Get Logged In User service (see section **Error! Reference source not found.**, **Error! Reference source not found.**) that may be used to test for a valid logon ticket.
- The X-Signature is mandatory for all POST operations.
- The sap-alias and sap-password parameters are required on the POST /api/zcur/users operation. Thereafter, the response SSO2 cookie can be optionally included in the other operations to provide user context to the services. (Note that partners that have not been provided with explicit username and password credentials may make use of the guest account login service (see section 4.2.1, API-WSR-US22: Login using a Guest User account).

4.2 Resource: (Guest Account) Login

The guest account login resource is used to authenticate and retrieve the SSO2 cookie or authentication token. Every HTTP interaction (that does not explicitly perform a logion operation) must include a valid SSO2 cookie otherwise a 401 error will result. The login resource supports the following operations:

ID	HTTP Verb	Path relative to : /coct/api/zcur-guest	Description
API-WSR-US22	GET	/login	This operation is used to login using a guest account, returning the SSO2 cookie needed for most other interactions..

Table 2: Operation for Resource: User relative to /api/zcur-guest

4.2.1 API-WSR-US22: Login using a Guest User account

HTTP Verb	GET
Relative API URL	/coct/api/zcur-guest/login
Header Parameters	{none}
Input	{none}
Input Schema	

Response	{
	login: successful
	}

Response Schema	{"login" :"successful"}

Table 3: API-WSR-US22 – Login using a Guest User account

4.3 Resource: Session

The only supported operation for the "session" resource is to get a session identifier for the purpose of and following the guidelines as described earlier in section 2.5 (page 12). The operations

ID	HTTP Verb	Path relative to : /coct/api/zsreq	Description
API-WSR-SE01	GET	/session	Retrieves a new session identifier for the X-Session header parameter. See section 2.4 (page 11) for guidelines on the use of the session identifier.

Table 4: Operations for Resource: Session

4.3.1 Session Operation API-WSR-SE01: Get Session Identifier

This service retrieves a session identifier, in the form of a 32-character Globally Unique Identifier (GUID), to be used for the X-Service parameter.

HTTP Verb	GET
URI	/coct/api/zsreq/session
Header Parameters	X-Service
Input Parameters	None

Response Model	{
	session_id (string): session identifier (a 32-character GUID)
	}

Response Schema	{"SessionId":"F6E9F4F4833B1ED49C8F9915F2A21F35"}
Return Codes / Messages	200 OK

Table 5: API-WSR-SE01 – Get Session Identifier

4.4 Resource: Configuration

The configuration resource delivers a number of services for fetching domain data and configuration data from the City's system that must be used in compiling the data to be posted through the API to the City's system.

Services are provided for on-line interaction with the City's systems that enable client applications retrieve the data in real-time. There are also services that can be used to retrieve the entire data sets so that these can be cached locally or centrally as desired. The domain and configuration data is very static so caching is entirely feasible. There should however be a mechanism put in place to refresh the cached data periodically as services are changed from time to time. A weekly refresh should be more than adequate.

It must also be noted that, if a service is changed from, for example, a general service class to an account-based service, any service requests submitted for that service with old (cached) data will be rejected with an error message due to the fact that the account number is not supplied.

The configuration resource supports the following operations:

ID	HTTP Verb	Path relative to : /coct/api/zsreq	Description
API-WSR-CN01	GET	/config/types	Retrieves a complete list of type codes and descriptions that constitutes the lookup list for the first of the two levels of hierarchical categorisation attributes for service requests (A.K.A group code and description)[1].
API-WSR-CN02	GET	/config/types/{type}	Retrieves the hierarchically related list of subtype codes and descriptions for a specific type (group of services) that constitutes the lookup list for the second of the two levels of hierarchical categorisation attributes for service requests (A.K.A service code and description). Note that these are the individual services for a specific group of services.
API-WSR-CN03	GET	/config/subtypes	Retrieves a complete list of subtype codes and descriptions for all available types[1]. This service is relevant only if this data is to be cached outside of the City's servers. If the data is not cached, service ID API-WSR-API-WSR-CN02 should be used *on change of the type code* to retrieve the subtypes for the selected type.
API-WSR-CN04	GET	/config	Retrieves the account requirement indicator for all services in all groups[1]. This service is relevant only if this data is to be cached outside of the City's servers. If the data is not cached, the service API-WSR-AC01 (described in section 4.5.1, API-WSR-AC01: Get Service Class, should be used with the selected type and subtype to retrieve the account requirement indicator for the selected combination.

Table 6: Operations for Resource: Configuration

[1] This data is very static and can be cached provided a mechanism is put in place for periodic invalidation and refresh of the cached data.

4.4.1 API-WSR-CN01: Get Types

This service retrieves the complete domain of the available type codes and descriptions. This is the domain of permissible values for the first of the two hierarchically-related levels used in the categorisation of services.

This domain can be cached using this service.

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/config/types
Header Parameters	X-Service X-Session
Input Parameters	None

Response	{
	type (string): a 4-character code identifying a specific grouping of services,
	name (string): a description of the service or grouping represented by the code
	}

Response Schema	[{"type" :1001,"name" :"Animal carcass removal"},{"type" :1047,"name" :"Burst water pipes, pressure and supply issues"},{"type" :1002,"name" :"City parks (Dumping and vagrancy)"},{"type" :1003,"name" :"City parks (Maintenance)"},{"type" :1004,"name" :"Electricity (Domestic and commercial supply)"},{"type" :1005,"name" :"Electricity (Equipment damage and exposure)"},{"type" :1007,"name" :"Electricity (Issues resulting from motor vehicle accidents)"},{"type" :1006,"name" :"Electricity (Meter readings)"}, …]
Return Codes / Messages	200 OK 500 Error in retrieving all type codes and descriptions. (Internal)

Table 7: API-WSR-CN01 – Get Types

4.4.2 API-WSR-CN02: Get Subtypes

This service retrieves the subtype codes and descriptions valid for a specific group of services (type code). This is the domain of permissible values for a specific type code for the second of the two hierarchically-related levels used in the categorisation of services.

This domain can be cached using API-WSR-CN03, described in section 4.4.3, API-WSR-CN03: Get All Subtypes.

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/config/types/{type}
Header Parameters	X-Service X-Session
Input Parameters	None

Response	{
	type (string): a 4-character code identifying a specific grouping of services,
	subtype (string): a four-character code identifying an individual service,
	name (string): a description of the service or grouping represented by the code
	}

Response Schema	[{"type" :1001,"subtype" :1001,"name" :"Cat"},{"type" :1001,"subtype" :1002,"name" :"Cow"},{"type" :1001,"subtype" :1003,"name" :"Dog - large"},{"type" :1001,"subtype" :1004,"name" :"Dog - small"},{"type" :1001,"subtype" :1005,"name" :"Horse or donkey"},{"type" :1001,"subtype" :1006,"name" :"Seal - large"},{"type" :1001,"subtype" :1007,"name" :"Seal - small"},{"type" :1001,"subtype" :1008,"name" :"Sheep"},{"type" :1001,"subtype" :1009,"name" :"Whale"}]
Return Codes / Messages	200 OK 500 Error in retrieving all sub-type codes and descriptions for type. (Internal)

Table 8: API-WSR-CN02 – Get Subtype

4.4.3 API-WSR-CN03: Get All Subtypes

This service retrieves the entire domain of subtype codes and descriptions for all service groups (type codes). This is the complete domain of permissible values for all type codes and must be filtered by type code to obtain the subset of values permissible for a specific type for the second of the two hierarchically-related levels used in the categorisation of services.

This service is used specifically for caching this domain. It is only relevant if the domain is to be cached otherwise the service API-WSR-CN02 (see section 4.4.2, API-WSR-CN02: Get Subtypes) is used to determine the permissible subtypes in an online environment.

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/config/subtypes
Header Parameters	X-Service X-Session
Input Parameters	None

Response	{
	type (string): a 4-character code identifying a specific grouping of services,
	subtype (string): a four-character code identifying an individual service,
	name (string): a description of the service or grouping represented by the code
	}

Response Schema	[{"type" :1001,"subtype" :1001,"name" :"Cat"},{"type" :1001,"subtype" :1002,"name" :"Cow"},{"type" :1001,"subtype" :1003,"name" :"Dog - large"},{"type" :1001,"subtype" :1004,"name" :"Dog - small"},{"type" :1001,"subtype" :1005,"name" :"Horse or donkey"},{"type" :1001,"subtype" :1006,"name" :"Seal - large"},{"type" :1001,"subtype" :1007,"name" :"Seal - small"},{"type" :1001,"subtype" :1008,"name" :"Sheep"},{"type" :1001,"subtype" :1009,"name" :"Whale"}]
Return Codes / Messages	200 OK 500 Error in retrieving all type codes and descriptions. (Internal)

Table 9: API-WSR-CN03 – Get All Subtypes

4.4.4 API-WSR-CN04: Get Service Configuration

This service retrieves the configuration defining the services (subtypes) that are account-based and those that are not. Account-based services are those that have the Boolean "account" flag set to true. Those with a false value are general services.

This service is used specifically for caching this configuration data. It is only relevant if the configuration is to be cached otherwise the account service API-WSR-AC01 (see section 4.5.1, API-WSR-AC01: Get Service Class) is used to determine the permissible subtypes in an interactive environment.

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/config
Header Parameters	X-Service X-Session
Input Parameters	None

Response	{
	type (string): a 4-character code identifying a specific grouping of services,
	subtype (string): a four-character code identifying an individual service,
	account_required (Boolean) : a logical value set to "true" if the service is account-based otherwise "false"
	}

Response Schema	[{"type" :1009,"subtype" :1003,"account" :""},{"type" :1037,"subtype" :1001,"account" :""},{"type" :1037,"subtype" :1002,"account" :""},{"type" :1037,"subtype" :1003,"account" :""},{"type" :1037,"subtype" :1004,"account" :""},{"type" :1037,"subtype" :1005,"account" :""},{"type" :1046,"subtype" :1000,"account" :""},{"type" :1047,"subtype" :1000,"account" :""},{"type" :1047,"subtype" :1001,"account" :"X"},{"type" :1047,"subtype" :1002,"account" :"X"},...]
Return Codes / Messages	200 OK 500 Error in retrieving all service class definitions. (Internal)

Table 10: API-WSR-CN04 – Get Service Configuration

4.5 Resource: Account

The account resource delivers operations specific to account-based services. An account-based service (as opposed to a general service) is one in which the location of the requested service is defined using the resident's municipal account number and where a valid municipal account must be provided in order to request the service.

It is imperative that the account number submitted for an account-based is a valid and current City municipal account number (otherwise the create service request operation will be rejected). This resource is used to determine whether an account number is mandatory as well as to validate and retrieve some details of an account number.

The configuration that describes services as account-based or otherwise can be cached using service API-WSR-CN04 (see section 4.4.4, API-WSR-CN04: Get Service Configuration), which would eliminate the need for using the service described here interactively.

The master data used to determine whether or not an account number is valid is too dynamic to be cached. This validation can therefore only be executed online. If an invalid account number is submitted, the create service request operation will fail with an error result.

ID	HTTP Verb	Path relative to : /coct/api/zsreq	Description
API-WSR-AC01	GET	/account/required?type={type}&subtype={subtype}	This operation (Get Service Type) returns a Boolean indicator set to true if the service defined by the URL parameters is an account-based service and false otherwise (indicating a general service).
API-WSR-AC02	GET	/account/{accountno}/validate	This operation (Get Account Details) returns a Boolean indicator that identifies if the account number submitted is valid and current and, if valid, some additional details about the account.

Table 11: Operations for Resource: Account

4.5.1 API-WSR-AC01: Get Service Class

This operation is used to determine where a specific service is an account-based service or not. The service in question is defined in the URL by the type and subtype codes for the service. It is possible to cache the configuration data that determines whether a service is account-based or not using service API-WSR-CN04 (see section 4.4.4, API-WSR-CN04: Get Service Configuration). If the data is not cached, this Get Service Class operation must be used to determine the service class.

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/account/required?type={type}&subtype={subtype}
Header Parameters	X-Service X-Session
Input Parameters	URL parameters: type(string): the type (group) code of the group containing the service in question subtype(string): the subtype (service) code of the specific service to be checked

Response	{
	account_required (Boolean) : a logical value set to "true" if the service is account-based otherwise "false"
	}

Response Schema	{"required" :true}
Return Codes / Messages	200 OK 500 Error in determining service class. (Internal)

Table 12: API-WSR-AC01 – Get Service Class

4.5.2 API-WSR-AC02: Get Account Details

This service is used to determine whether or not a specific account number is valid and current and, if it is, to return some details about the account, notably the street address of the property to which the account is linked.

The master data used to determine whether or not an account number is valid is too dynamic to be cached. This validation can therefore only be executed online. If an invalid account number is submitted, the create service request operation will fail with an error result.

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/account/{accountno}/validate
Header Parameters	X-Service X-Session
Input Parameters	None

Response	{
	invalid_account (Boolean): set "true" if the account number provided is not a valid City municipal account number, "false" otherwise,
	account_number (string): a City municipal account number,
	address (string): the comma-delimited address defining the location of a property or service request,
	latitude (string): the lattitude of the location of the property (in decimal degrees),
	longitude (string): the longitude of the location of the property (in decimal degrees)
	}

Response Schema	{"invalid_account" :false,"account_number" :"200274203","address" :"24 ALICANTE AVENUE / MILNERTON","latitude" :"","longitude" :""}
Error 500 Messages	500 Error in retrieving all type codes and descriptions. (Internal)

Table 13: API-WSR-AC01 – Get Account Details

4.6 Resource: Service Request

This resource is used to create service requests with the data captured on the consumer's application and to query the details and current state of a request given the reference number.

When creating a service request, the data that is submitted must conform to the requirements as set out in this document. The data will be validated before posting and will be rejected with errors if it does not conform.

ID	HTTP Verb	Path relative to : /coct/api/zsreq	Description
API-WSR-R01	POST	/sr	This operation (Create Service Request) is used to create a service request in the City's environment with the data captured on the mobile application.
API-WSR-R02	GET	/sr/{srrefno}	This operation (Get Service Request Detail) is used to query the details and current state of a service request in the City's environment give the reference number of the specific request.

Table 14: Operations for Resource: Service Request

4.6.1 API-WSR-SR01: Create Service Request

This operation is used to create a service request in the City's system. The data submitted is checked for validity and, if there are errors, the request will be rejected with an error message. If the data is valid then a service request will be created and the reference number and details of that request will be returned to the caller.

This is a POST operation. It requires an HMAC to be computed (see section 2.3, Message Authentication (HMAC)and submitted in the header parameter X-Signature.

HTTP Verb	POST
Relative API URL	/coct/api/zsreq/sr
Header Parameters	X-Service X-Session X-Signature

Input	{
	type (string): the four-character code of the service group containing the selected service,
	subtype (string): the four-character code of the specific service selected,
	message (string): the text describing the requested service as entered by the user,
	address (string): the comma-delimited address of the location of the service request,
	latitude (string): the lattitude of the location of the property (in decimal degrees),
	longitude (string): the longitude of the location of the property (in decimal degrees)
	email (string, optional): valid email address (required if commpref is "email"),
	telephone (string): valid local mobile telephone number,
	comm (string) = ["INT", "SMS"]: communication preference,
	username (string, optional): CCT username assigned to users registered on the City'solution
	}
	{
	username (string): a 12-character internal name assigned to a user's profile,
	firstName: the first name of the contact person {maximum 40 characters},
	lastname (string): the last name of the contact person {maximum 40 characters},
	account_number (string): a City municipal account number,
	type (string): a 4-character code identifying a specific grouping of services,
	subtype (string): a four-character code identifying an individual service,
	address (string): the comma-delimited address defining the location of a property or service request,
	street_number (string): the street number component of the location of the service request,
	street (string): the street name component of the location of the service request,
	suburb (string): the suburb name component of the location of the servcie request,
	telephone (string): valid local mobile telephone number,
	email (string): an email address at which the contact person can be reached,
	message (string): a text description including further details of the requested service and feedback,
	latitude (string): the lattitude in decimal degrees of a geospatial point,
	longitude (string): the longitude in decimal degrees of a deospatial point,
	comm (string) = ["EMAIL", "SMS", "NONE"]: the preferred correspondence method for the service request feedback
	filename(string): the name of the file attached
	filetype(string): the mimetype type of the file (from the list of permissable types:
	{
	application/pdf,
	application/msword,
	application/vnd.openxmlformats-officedocument.wordprocessingml.document,
	application/vnd.ms-excel,
	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet,
	application/vnd.ms-powerpoint,
	application/vnd.openmlformats-officeoducment.presentationml.presentation,
	image/jpeg,
	image/png,
	image/gif,
	image/tiff,
	image/bmp
	}
	filecontent(string): Base64 encoded file content
	}

Input Schema	{"username":"","firstName":"John","lastName":"Doe","account_number":"","type":"1001","subtype":"1009", "address":"0A, Beach Road, Muizenberg","street_number":"0A","street":"Beach Road","suburb":"Muizenberg","telephone":"0835559876","email":"j.doe@gmail.com","message":"Please collect the whale carcass from Muizenberg beach before the wind direction changes.","latitude":"-34.108038128850701298","longitude":"18.471525907516479492","comm":"EMAIL"}

Response	{
	reference_number (string): the reference number of the service request,
	category (string): a description of the requested service category,
	message (string): a text description including further details of the requested service and feedback
	}

Response Schema	{"reference_number" :"000010011101","category" :"Animal carcass removal - Whale","description" :"","message" :"Please collect the whale carcass from Muizenberg beach before the wind direction changes."}

Table 15: API-WSR-SR01 – Create Service Request

4.6.2 API-WSR-SR02: Get Service Request Detail

This operation is used to retrieve the current state of a service request given the service request number.

Service requests have a defined lifecycle that is governed by a progression of statuses. During the lifecycle of a service request it is common for the City's staff to add comments to the service request describing findings and aspects such as transfer of work. The service described here makes these changes visible to the customer.

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/sr/{referenceno}
Header Parameters	X-Service X-Session X-Signature
Input Parameters	None

Response	{
	reference_number (string): the reference number of the service request,
	type (string): a 4-character code identifying a specific grouping of services,
	subtype (string): a four-character code identifying an individual service,
	description (string): the short description of the service reqeust,
	status (string): a description of the current status of the request,
	message (string): a text description including further details of the requested service and feedback,
	time_created (string, "hh:mm:ss"): the time at which the request was created,
	date_created (string, "yyyymmdd"): the date on which the request was created,
	description (string): a short description of the requested service
	}

Response Schema	{"reference_number" :"000010011101","type" :"1001","subtype" :"1002","status" :"Outstanding notificationAll Tasks Completed","message" :" Please collect the whale carcass from Muizenberg beach before the wind direction changes. 28.11.2014 11:20:13 Gareth Dohne (SC_GD1) The carcass has been successfully removed from the beach and the surrounding area cleaned up.","time_created" :"10:08:59","date_created" :"20141128","description" :"Animal carcass removal - Whale"}

Table 16: API-WSR-SR01 – Get Service Request Detail

4.7 Resource: (Generic) Users

This Users resource is used to authenticate a user using basic authentication (explicit username and password) as well as to retrieve the details of authenticated user. register and manage users in the City's environment. The use of this resource is optional.

The generic Users resource described here is intended for use in the City's application and has limited no benefit at this point for third-party consumers of the City's API who are currently making use of the guest account login service (see section 4.2.1, API-WSR-US22: Login using a Guest User account).

The operation to retrieve the currently authenticated user's details may be useful for third-parties as a simple mechanism to determine whether or not the SSO2 ticket currently in use has expired. (If it has expired, this or almost any other service will return a 401 error). This Users resource supports the following operations:

ID	HTTP Verb	Path relative to : /coct/api/zcur	Description
API-WSR-US30	POST	/users	This operation is used to login a user and return an SSO2 Token (as a cookie)
API-WSR-US31	GET	/users/me	This operation is used to get the currently logged in user (must have a valid SSO2 Token). If this request fails with either HTTP code 500 or 401, then the token is invalid and a login operation needs to occur again in order to refresh the token.

Table 17: Operation for Resource: User relative to /api/zcur

4.7.1 API-WSR-US30: Login

HTTP Verb	POST
Relative API URL	/coct/api/zcur/users
Header Parameters	X-Service X-Session sap-alias sap-password
Input	{user alias and password to passed into the headers above}
Input Schema	
Response	SSO2 Cookie is returned
Response Schema	SSO2 Cookie is returned

Table 18: API-WSR-US30 – Login

4.7.2 API-WSR-US31: Get currently logged in User

HTTP Verb	GET
Relative API URL	/coct/api/zcur/users
Header Parameters	X-Service X-Session
Input	{user alias and password to passed into the headers above}
Input Schema	

Response	{
	firstname (string): the first name of the contact person {maximum 40 characters},
	lastname (string): the last name of the contact person {maximum 40 characters},
	email (string): a valid email address for the contact person,
	pref_comm (string) = ["E", "S"]: the preferred correspondence method for the service request feedback,
	title (string) = ["MR", "MRS", "MS"],
	middle_name (string): the middle name of the contact person,
	telephone (string): the telephone number of the contact person,
	mobile_number (string): the mobile number of the contact person,
	fax (string): the fax number of the contact person,
	pref_lang (string) = ["E"],
	sms (string) = ["", "X"],
	initials (string): the initials of the contact person
	}

Response Schema	

Table 19: API-WSR-US31 – Get currently logged in user

4.8 Resource: (Guest Account) Users *DEPRECATED*

The Users resource on the guest account API offered the ability to create new user profiles and change passwords for users. The registration model used in this service is obsolete and requires re-design. This resource was intended to fulfil a need to register individual users from the City's own application. There is little benefit in individual user management on the City's database for a third-party API consumer so, the fact that this resource is not available should not have any negative impact on third-parties seeking to consume the City's service request API. The use of this resource was always intended to be optional.

User registration and management is useful in that it delivers context and additional functionality for individuals who make regular use of the City's services (directly). The Users resource supports the following (deprecated) operations:

ID	HTTP Verb	Path relative to : /coct/api/zcur-guest	Description
API-WSR-US20	POST	/users	This operation is used to create a user profile (register a user) on the City's system.
API-WSR-US21	PUT	/users	This operation is used to reset the user's password.

Table 20: Deprecated Operation for Resource: Users relative to /api/zcur-guest

4.8.1 API-WSR-US20: Register a new user

HTTP Verb	POST
Relative API URL	/coct/api/zcur-guest/users
Header Parameters	X-Service X-Session X-Signature

Input	{
	firstname (string): the first name of the contact person {maximum 40 characters},
	lastname (string): the last name of the contact person {maximum 40 characters},
	email (string): a valid email address for the contact person,
	pref_comm (string) = ["E", "S"]: the preferred correspondence method for the service request feedback,
	idnumber (string): the ID Number of the contact person,
	idtype (string) = ["001" (SA ID), "003" (Passport)],
	title (string) = ["MR", "MRS", "MS"],
	idcountry (string) = ["ZA"],
	middle_name (string): the middle name of the contact person,
	telephone (string): the telephone number of the contact person,
	mobile_number (string): the mobile number of the contact person,
	fax (string): the fax number of the contact person,
	pref_lang (string) = ["E"],
	sms (string) = ["", "X"],
	initials (string): the initials of the contact person,
	user_alias (string): the chosen username of the contact person,
	password (string): the chosen password of the contact person,
	confirm_password (string): the chosen password (repeated) of the contact person
	}

Input Schema	

Response	{
	userid :CU0000002109,
	password :qwerty
	}

Response Schema	{"userid" :"CU0000002109","password" :"qwerty"}

Table 21: API-WSR-US20 – Register a new user

4.8.2 *API-WSR-US21: Reset user password*

HTTP Verb	PUT
Relative API URL	/coct/api/zcur-guest/users
Header Parameters	X-Service X-Session X-Signature

Input	{
	email (string): a valid email address for the contact person {either email or id_number should be populated},
	id_number (string): the ID Number of the contact person {either email or id_number should be populated},
	user_alias (string): the chosen username of the contact person,
	password (string): the chosen password of the contact person,
	confirm_password (string): the chosen password (repeated) of the contact person
	}

Input Schema	

Response	{
	userid :CU0000002109,
	password :qwerty
	}

Response Schema	{"userid" :"CU0000002109","password" :"qwerty"}

Table 22: API-WSR-US21 – Reset a user's password

4.9 Resource: User *DEPRECATED*

The User resource was intended to be used to register and manage users and their details in the City's environment. The use of this resource was always intended to be optional and was aimed specifically at deployment in the City's application. The resource would be of limited benefit to third-party API consumers who would more likely have their own form of user accounts and related master data.

The registration model implemented using this resource requires a re-design, which is why the use of this is now deprecated. A replacement resource / version may be implemented in future should demand and capacity dictate. API consumers will be advised of any changes.

The objective with user registration and master data management to provide user-specific context and additional functionality for individuals who make regular use of the service request solution by pre-populating master data or providing selection lists specific to a user where appropriate. The User resource supported the following (now deprecated) operations:

ID	HTTP Verb	Path relative to : /coct/api/zsreq	Description
API-WSR-US01	POST	/user	This operation is used to create a user profile (register a user) on the City's system. DEPRECATED.
API-WSR-US02	GET	/user/{username}	This operation is used to retrieve the user's profile from the City' system. DEPRECATED.
API-WSR-US03	PUT	/user/{username}	This operation is used to change the user's profile on the City' system. DEPRECATED.

ID	HTTP Verb	Path relative to : /coct/api/zsreq	Description
API-WSR-US04	GET	/user/{username}/numbers	This operation is used to get the list of mobile device numbers currently associated with the user {username}. DEPRECATED.
API-WSR-US05	POST	/user/{username}/numbers	This operation is used to add a new mobile number to the user's mobile device number profile. DEPRECATED.
API-WSR-US06	PUT	/user/{username}/numbers/{number}	This operation is used to change the active number for the user {username} to the number {number}. DEPRECATED.
API-WSR-US07	DELETE	/user/{username}/numbers/{number}	This operation is used to delete the mobile number {number} for the user {username}. DEPRECATED.
API-WSR-US08	GET	/user/me/accounts	This operation is used to retrieve the list of municipal account numbers associated with the user (request must contain a valid SSO2 cookie).
API-WSR-US09	POST	/user/me/accounts	This operation is used to add a municipal account number to the list of numbers associated with the user (request must contain a valid SSO2 cookie).
API-WSR-US10	DELETE	/user/me/accounts/{accountno}	This operation is used to delete the account number {accountno} from the list of municipal account numbers associated with the user (request must contain a valid SSO2 cookie).
API-WSR-US11	POST	/user/{username}/pin	This operation is used to reset the PIN code used for authenticating the user {username}. DEPRECATED.
API-WSR-US12	POST	/user/login	This operation is used to authenticate a user on the City's system. DEPRECATED.
API-WSR-US13	GET	/user/me/sr	This operation is used to retrieve the user's service requests (request must contain a valid SSO2 cookie).

Table 23: Deprecated Operations for Resource: User

4.9.1 API-WSR-US01: Create User Profile DEPRECATED

This service is used to create a new user profile on the City's environment. This is equivalent to registering a user. The data provided

HTTP Verb	POST
Relative API URL	/coct/api/zsreq/user
Header Parameters	X-Service X-Session X-Signature

Input	{
	mobile (string): a valid mobile (SMS-capable) telephone number,
	pin (string): an authentication code / password {maximum of 10 digits},
	firstname (string): the first name of the contact person {maximum 40 characters},
	lastname (string): the last name of the contact person {maximum 40 characters},
	email (string): a valid email address for the contact person,
	comm (string) = ["EMAIL", "SMS", "NONE"]: the preferred correspondence method for the service request feedback
	}

Input Schema	{"mobile":"0835559876","pin":"1234","firstname":"John","lastname":"Doe","email":"j.doe@gmail.com","comm":"EMAIL"}

Response	{
	username (string): a 12-character internal name assigned to a user's profile,
	pin (string): an authentication code / password {maximum of 10 digits},
	lastname (string): the last name of the contact person {maximum 40 characters},
	firstname (string): the first name of the contact person {maximum 40 characters},
	email (string): a valid email address for the contact person,
	active_mobile (string): the currently active mobile number on a user's profile,
	default_account (string): the default municipal account number on a user's profile,
	comm (string) = ["EMAIL", "SMS", "NONE"]: the preferred correspondence method for the service request feedback
	}

Response Schema	{"username" :"CU0000002115","lastname" :"Doe","firstname" :"John","email" :"j.doe@GMAIL.COM","active_mobile" :"","default_account" :"","comm" :"EMAIL"}

Table 24: API-WSR-US01 – Create User Profile

4.9.2 API-WSR-US02: Get User Profile DEPRECATED

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/user/{username}
Header Parameters	X-Service X-Session
Input Parameters	None

Response	{
	username (string): a 12-character internal name assigned to a user's profile,
	lastname (string): the last name of the contact person {maximum 40 characters},
	firstname (string): the first name of the contact person {maximum 40 characters},
	email (string): a valid email address for the contact person,
	active_mobile (string): the currently active mobile number on a user's profile,
	default_account (string): the default municipal account number on a user's profile,
	comm (string) = ["EMAIL", "SMS", "NONE"]: the preferred correspondence method for the service request feedback
	}

Response Schema	{"username" :"CU00000002114","lastname" :"Doe","firstname" :"John","email" :"j.doe@GMAIL.COM","active_mobile" :"0835559876","default_account" :"","comm" :"EMAIL"}

Table 25: API-WSR-US02 – Get User Profile

4.9.3 API-WSR-US03: Change User Profile DEPRECATED

HTTP Verb	PUT
Relative API URL	/coct/api/zsreq/user/{username}
Header Parameters	X-Service X-Session X-Signature

Input	{
	firstname (string): the first name of the contact person {maximum 40 characters},
	lastname (string): the last name of the contact person {maximum 40 characters},
	email (string): a valid email address for the contact person,
	comm (string) = ["EMAIL", "SMS", "NONE"]: the preferred correspondence method for the service request feedback
	}

Input Schema	{"firstname":"John","lastname":"Doe","email":"j.doe@GMAIL.COM","comm":"EMAIL"}

Response	{
	mandt (string): an internal value of no interest to external consumers,
	username (string): a twelve-character internal name assigned to the user profile',
	pin (string): the user's authentication code,
	firstname (string): the first name of the user,
	lastname (string): the last name of the user,
	email (string): the email address of the user,
	email_u (string): the user's email address in uppercase for searching,
	comm_type (numeric) = [1, 2]: an integer indicating the user's preferred feedback channel,
	}

Response Schema	{"mandt" :"050","username" :"CU00000002112","pin" :"1234","lastname" :"Dick","firstname" :"Gareth","email" :"moby.dick@gmail.com","email_u" :"moby.dick@gmail.com","comm_type" :1}

Table 26: API-WSR-US03 – Change User Profile

4.9.4 API-WSR-US04: Get User's Mobile Numbers DEPRECATED

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/user/{username}/numbers
Header Parameters	X-Service X-Session
Input Parameters	None

Response	[
	{
	username (string): a twelve-character internal name assigned to the user profile',
	mobile (string): the mobile telephone number for which the user profile is to be created,
	mobile_u (string): the mobile telephone in searchable form (no punctuation),
	active (Boolean): a flag set if this is the user's default mobile number, false otherwise
	}
	]

Response Schema	[{"username" :"CU0000002107","mobile" :"0836250175","mobile_u" :"","active" :false},{"username" :"CU0000002107","mobile" :"0836251111","mobile_u" :"","active" :false},{"username" :"CU0000002107","mobile" :"0836251112","mobile_u" :"","active" :true}]

Table 27: API-WSR-US04 – Get User's Mobile Numbers

4.9.5 API-WSR-US05: Add User's Mobile Number DEPRECATED

HTTP Verb	POST
Relative API URL	/coct/api/zsreq/user/{username}/numbers
Header Parameters	X-Service X-Session X-Signature

Input	{
	mobile (string): the mobile telepehone number for which the user profile is to be created,
	active (Boolean): a flag set if this is the user's default mobile number, false otherwise
	}

Input Schema	{"mobile":"0836251113","active":" "}

Response	{
	mandt (string): an internal value of no interest to external consumers,
	username (string): a twelve-character internal name assigned to the user profile',
	mobile (string): the mobile telephone number for which the user profile is to be created,
	mobile_u (string): the mobile telephone in searchable form (no punctuation),
	active (Boolean): a flag set if this is the user's default mobile number, false otherwise
	}

Response Schema	{"mandt" :"","username" :"CU0000002107","mobile" :"0836251113","mobile_u" :"0836251113","active" :""}

Table 28: API-WSR-US05 – Add User's Mobile Number

4.9.6 API-WSR-US06: Change User's Active Mobile Number DEPRECATED

HTTP Verb	PUT
Relative API URL	/coct/api/zsreq/user/{username}/numbers/{mobileno}
Header Parameters	X-Service X-Session
Input Parameters	None

Response	{
	mandt (string): an internal value of no interest to external consumers,
	username (string): a twelve-character internal name assigned to the user profile',
	mobile (string): the mobile telephone number for which the user profile is to be created,
	mobile_u (string): the mobile telephone in searchable form (no punctuation),
	active (Boolean): a flag set if this is the user's default mobile number, false otherwise
	}

Response Schema	{"username" :"CU00000002114","mobile_number" :"0835559874","active" :true}

Table 29: API-WSR-US06 – Change User's Active Mobile Number

4.9.7 API-WSR-US07: Delete User's Mobile Number DEPRECATED

HTTP Verb	DELETE
Relative API URL	/coct/api/zsreq/user/{username}/numbers/{mobileno}
Header Parameters	X-Service X-Session
Input Parameters	None
Response	HTTP 200

Table 30: API-WSR-US07 – Delete User's Mobile Number

4.9.8 API-WSR-US08: Get User's Account Numbers DEPRECATED

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/user/me/accounts
Header Parameters	X-Service X-Session
Input Parameters	None`

Response	[
	{
	username (string): a twelve-character internal name assigned to the user profile',
	account_number (string): a municipal account number with the City
	}
	]

Response Schema	[{"username" :"CU00000002107","account_number" :"000200274092"},{"username" :"CU00000002107","account_number" :"000200274203"},{"username" :"CU00000002107","account_number" :"200271223"},{"username" :"CU00000002107","account_number" :"200274226"}]

Table 31: API-WSR-US08 – Get User's Account Numbers

4.9.9 API-WSR-US09: Add User's Account Number DEPRECATED

HTTP Verb	POST
Relative API URL	/coct/api/zsreq/user/me/accounts
Header Parameters	X-Service X-Session X-Signature

Input	{
	account_number (string): a municipal account number with the City
	}

Input Schema	{"account_number":"000200274203"}

Response	{
	"mandt" :"",
	"username" :"CU0000002107",
	"account_number" :"000200274203"
	}

Response Schema	{"mandt" :"","username" :"CU0000002107","account_number" :"000200274203"}

Table 32: API-WSR-US09 – Add User's Account Number

4.9.10 API-WSR-US10: Delete User's Account Number DEPRECATED

HTTP Verb	DELETE
Relative API URL	/coct/api/zsreq/user/me/accounts/{accountno}
Header Parameters	X-Service X-Session
Input Parameters	None
Response	HTTP 200

Table 33: API-WSR-US10 – Delete User's Account Number

4.9.11 API-WSR-US11: Change User's Authentication PIN Code DEPRECATED

HTTP Verb	POST
Relative API URL	/coct/api/zsreq/user/{username}/pin
Header Parameters	X-Service X-Session X-Signature
Input	
Input Schema	
Response	
Response Schema	

Table 34: API-WSR-US11 – Change User's Authentication PIN Code

4.9.12 API-WSR-US12: Authenticate User DEPRECATED

HTTP Verb	POST
Relative API URL	/coct/api/zsreq/user/login
Header Parameters	X-Service X-Session X-Signature
Input	
Input Schema	

Response	{
	"username" :"CU0000002109",
	"lastname" :"Dick",
	"firstname" :"Moby",
	"email" :"moby.dick@gmail.com",
	"primary_mobile" :"0825551000",
	"default_account" :"",
	"comm_email" :"true",
	"comm_mobile" :"false"
	}

Response Schema	{"username" :"CU0000002109","lastname" :"Dick","firstname" :"Moby","email" :"moby.dick@gmail.com","primary_mobile" :"0825551000","default_account" :"","comm_email" :"true","comm_mobile" :"false"}

Table 35: API-WSR-US12 – Authenticate User

4.9.13 API-WSR-US13: Get User's Service Requests DEPRECATED

HTTP Verb	GET
Relative API URL	/coct/api/zsreq/user/me/sr
Header Parameters	X-Service X-Session X-Signature
Input	
Input Schema	

Response	[
	{
	username :CU0000002109,
	lastname :Dick,
	firstname :Moby,
	email :moby.dick@gmail.com,
	primary_mobile :0825551000,
	default_account :,
	comm_email :true,
	comm_mobile :false
	}
	]

Response Schema	{"username" :"CU0000002109","lastname" :"Dick","firstname" :"Moby","email" :"moby.dick@gmail.com","primary_mobile" :"0825551000","default_account" :"","comm_email" :"true","comm_mobile" :"false"}

Table 36: API-WSR-US13 – Get User's Service Requests

5 Quality Assurance & Production System Hostnames

There are two platforms available to consumers of this interface, a quality assurance system and the live, production system. All development and testing must be carried out against the City's quality assurance system. Final acceptance testing must be carried out in collaboration with the City so that the results can be confirmed before approval will be granted to take the solution into our live environment.

Every effort must be made to ensure that test and junk data is not created in the City's live system as this could have a significantly negative impact on the City's capacity to deliver services. Permission to use the interface will be rescinded if this rule is violated.

Communication with the City's systems must be encrypted through the use of SSL and HTTPS protocol. (HTTP protocol is not supported.) The hostnames for the platforms are as follows:

Quality Assurance System	https://qaeservices1.capetown.gov.za
Live Production System	https://eservices1.capetown.gov.za

Table 37: Quality Assurance & Production System Hostnames

The various REST operations are accessed by appending the API URL to the above hostnames (for example, https://qaeservices1.capetown.gov.za/coct/api/zsreq/session).

Appendix A: Sample Screen Flows

A.1 Service Menu

A.1.1 Before Registration / Login

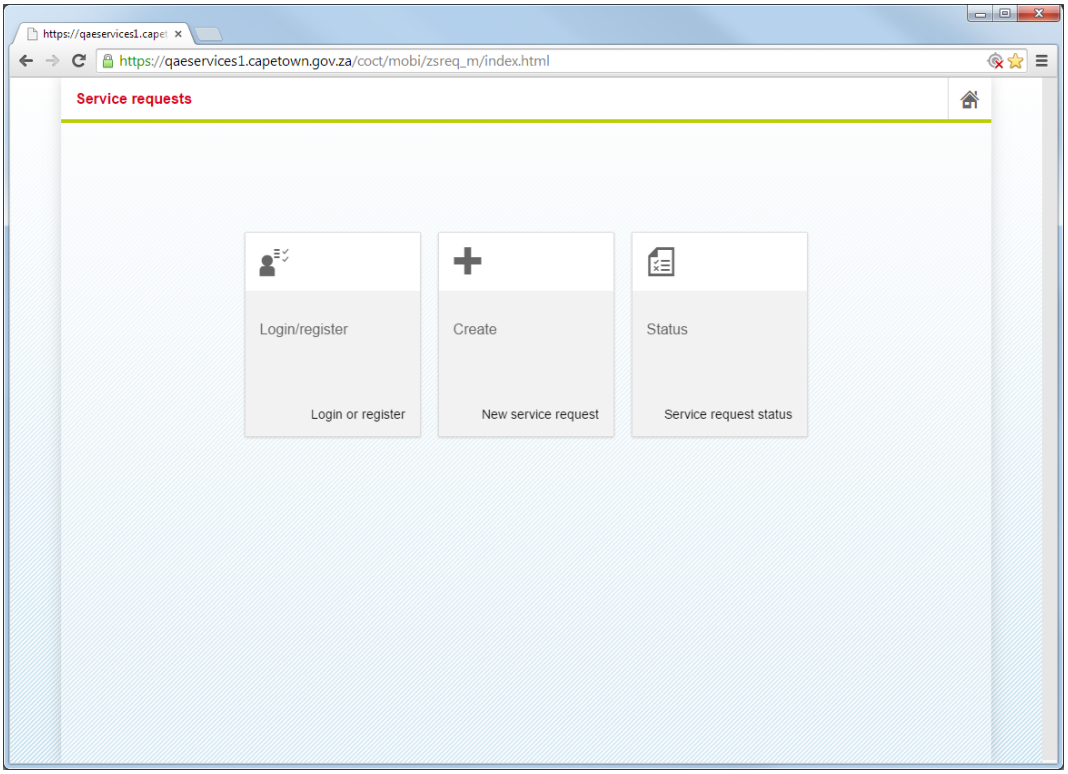

Figure 2: Service Menu with no Login

A.1.2 Registered User

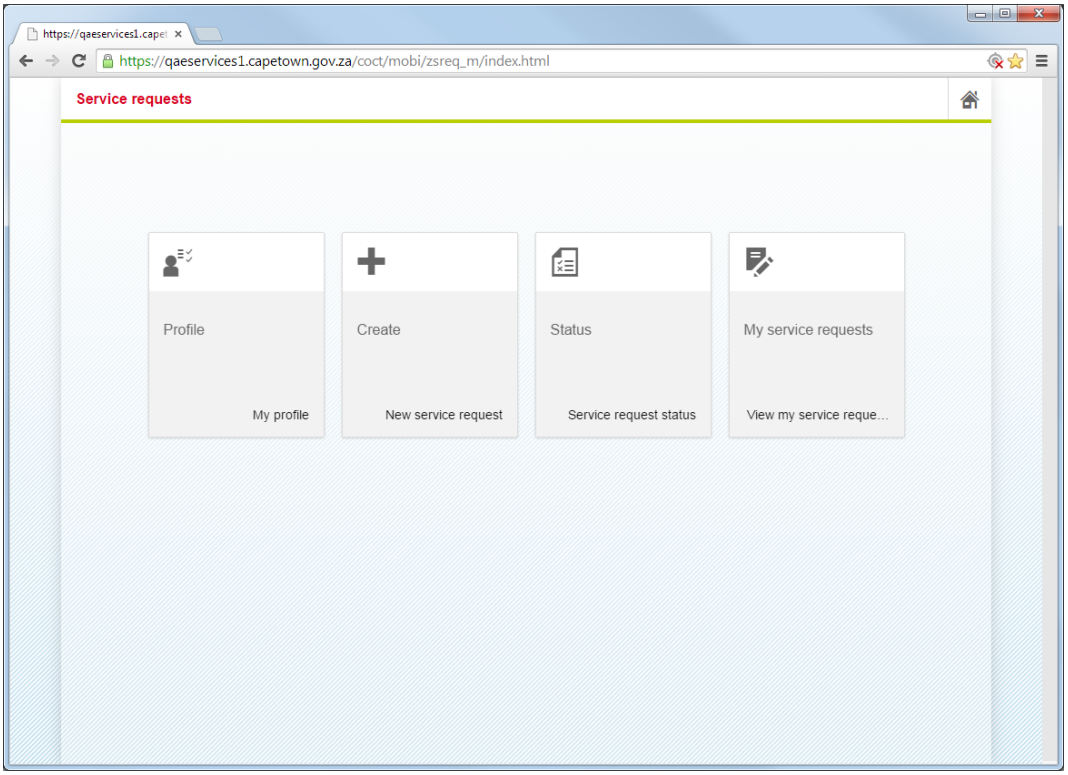

Figure 3: Service Menu for Logged In User

A.2 Service Request Create

A.2.1 Create SR for a General Service

Service Request Categorisation

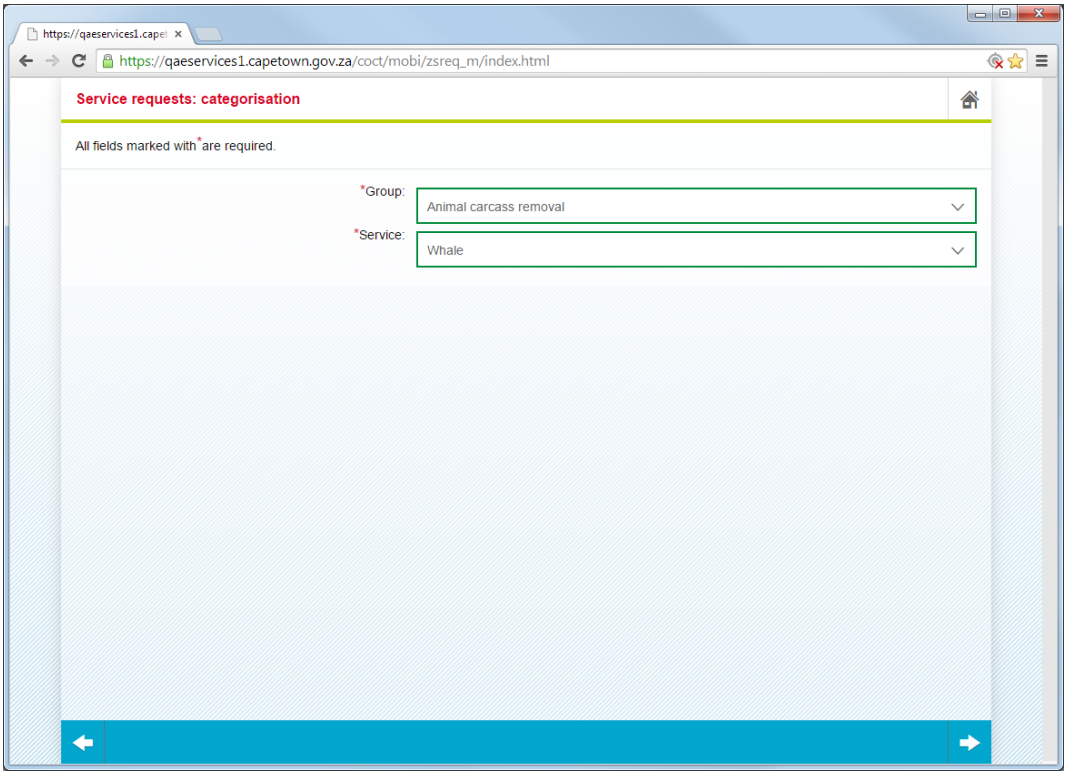

Figure 4: General Service – Categorise Request

Description of Requested Service

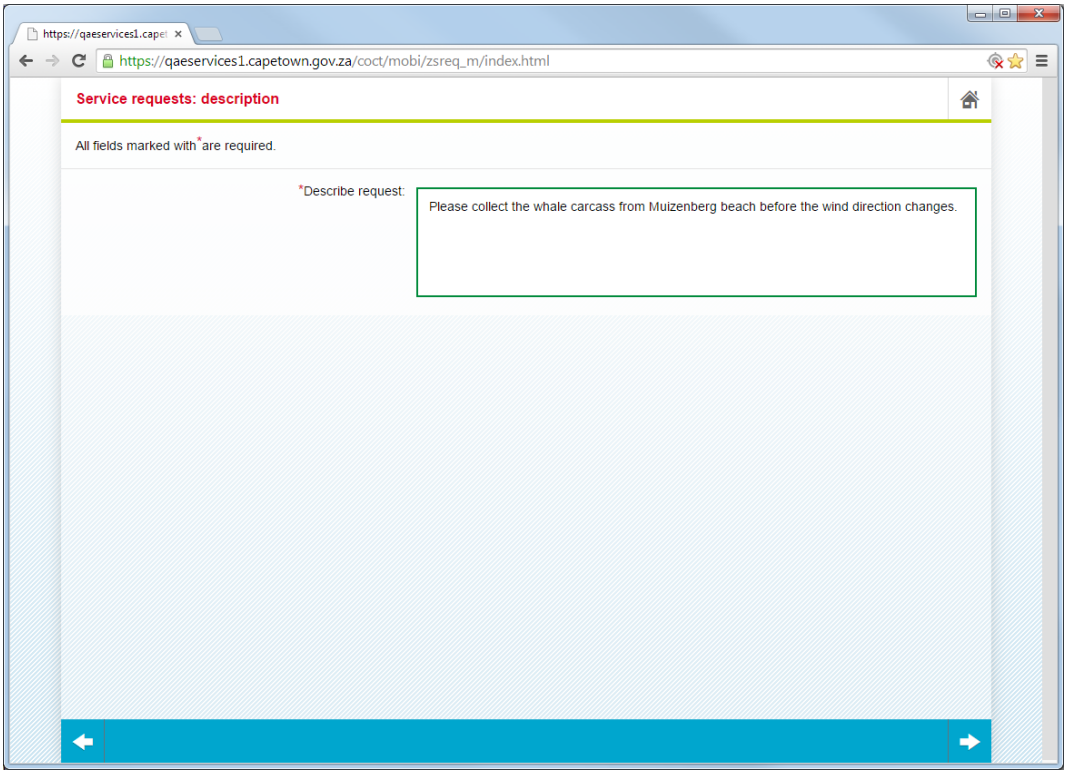

Figure 5: General Service – Description of Service

Location of Requested Service

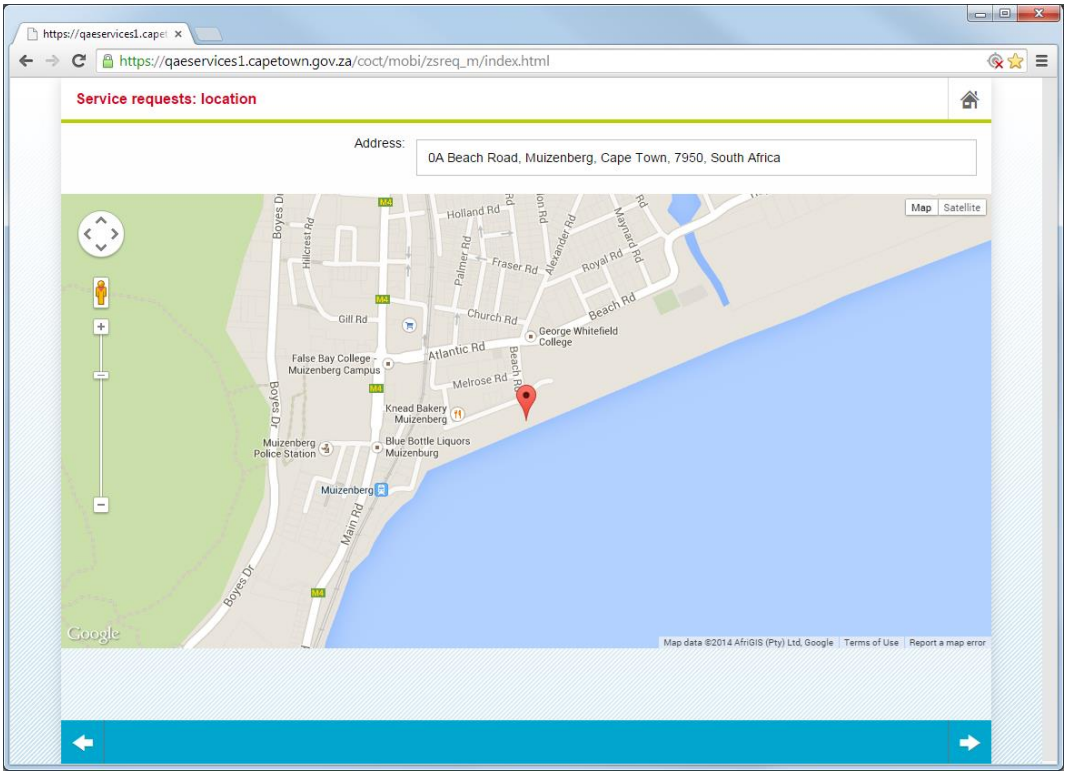

Figure 6: General Service – Location of Requested Service

Confirmation of Address

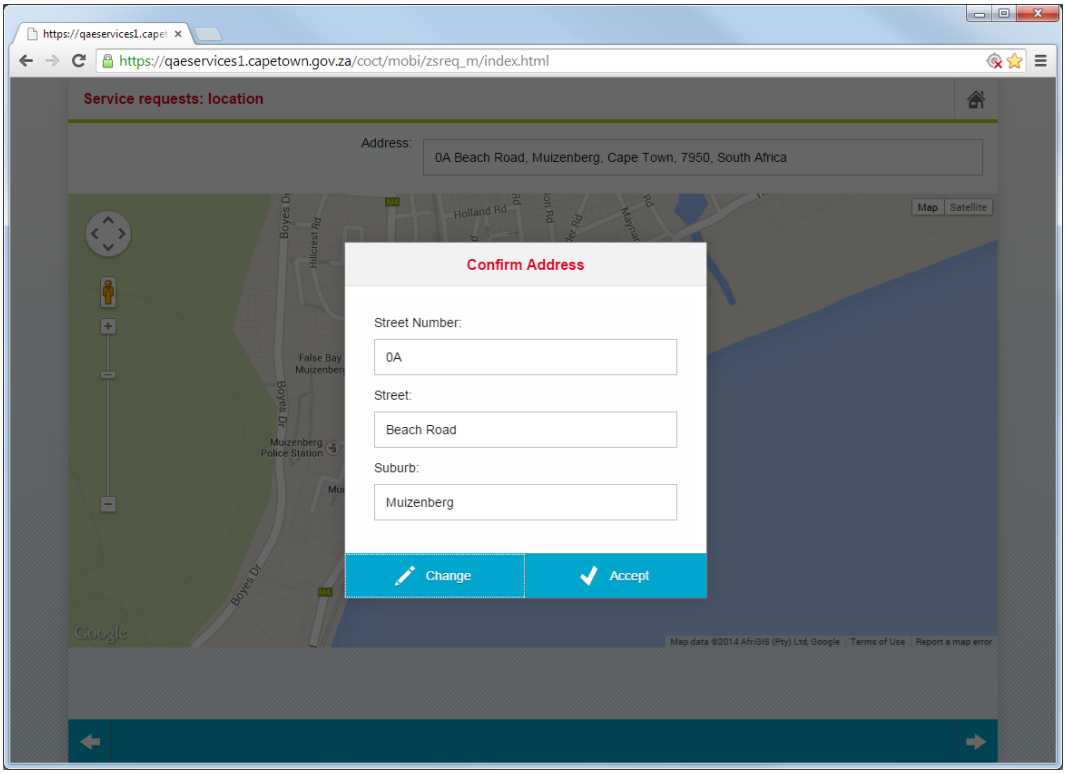

Figure 7: General Service – Confirmation of Address

Contact Details

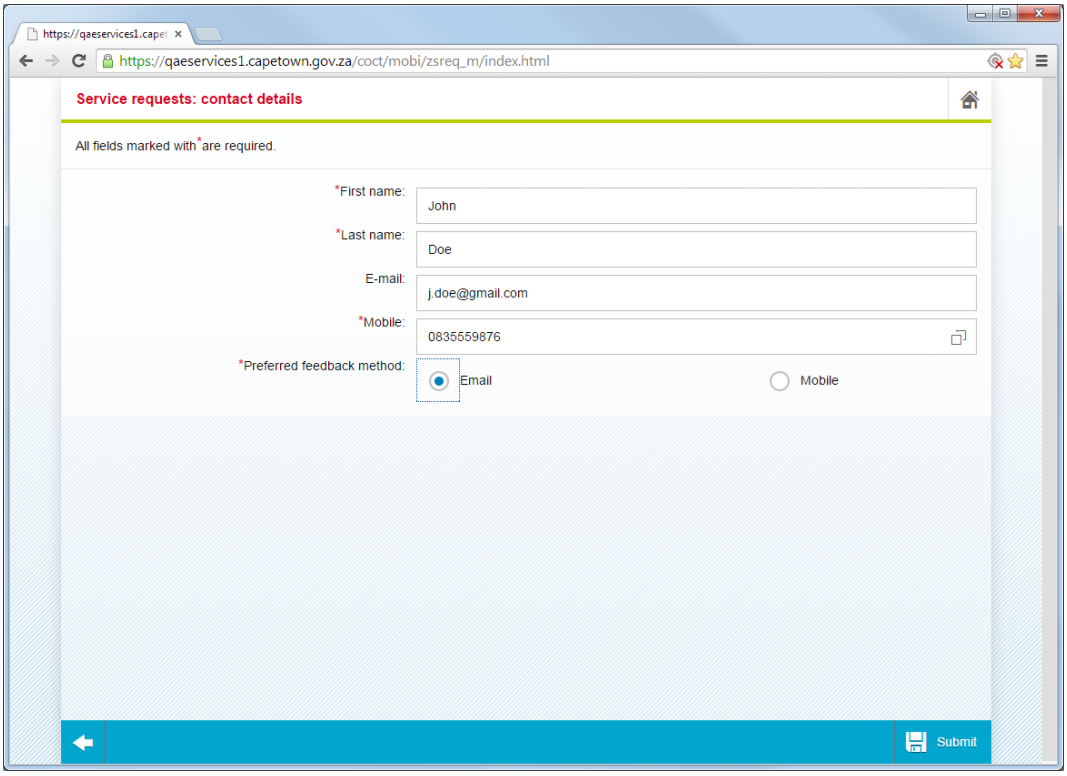

Figure 8: General Service – Contact Details

Confirmation of Service Request Created

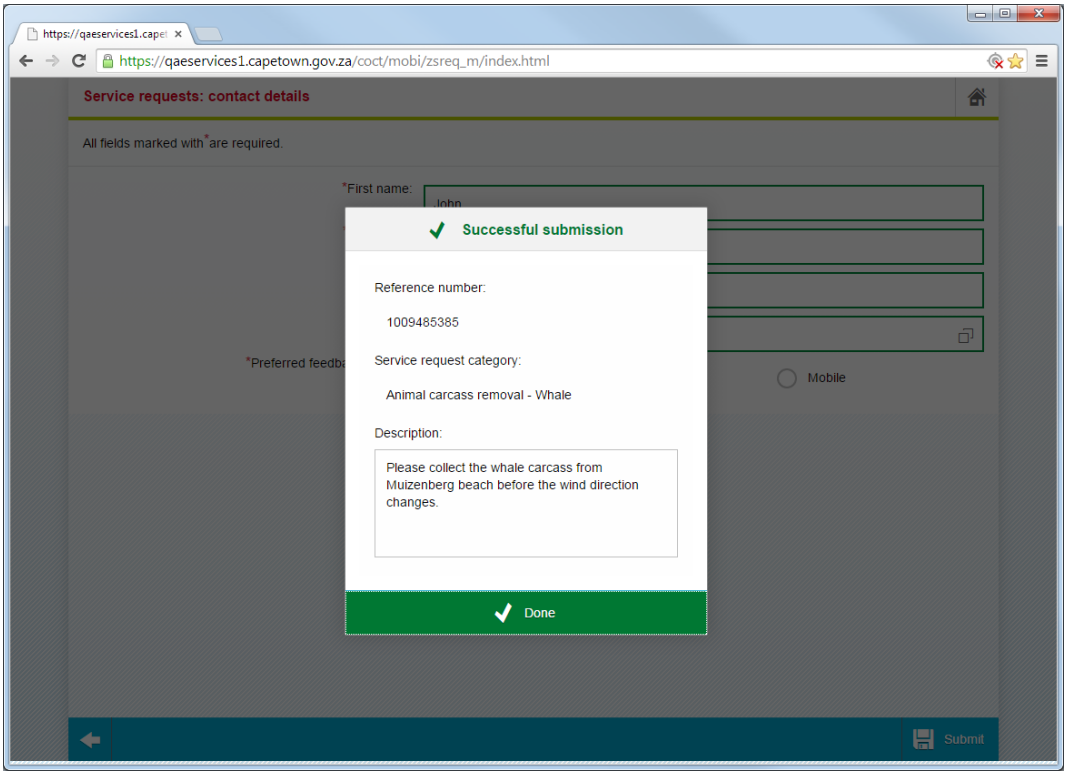

Figure 9: General Service – Confirmation of Request

A.2.2 Create SR for an Account-Based Service

Service Request Categorisation

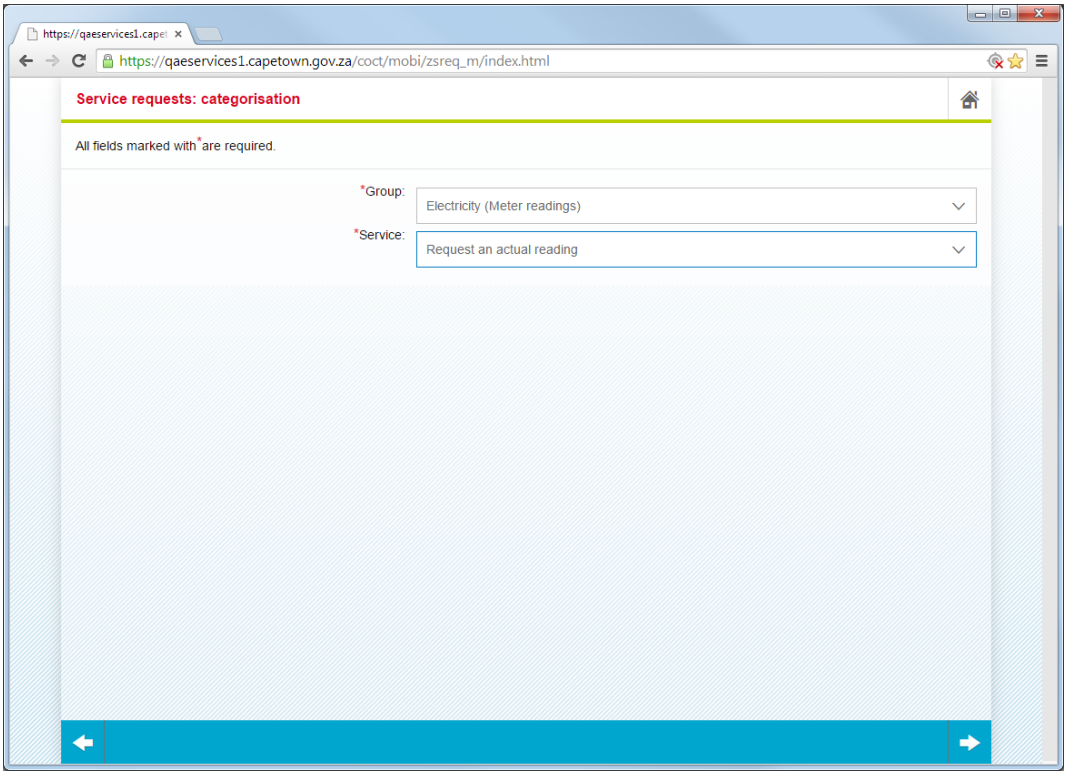

Figure 10: Account-Based Service – Categorise Request

Description of Requested Service

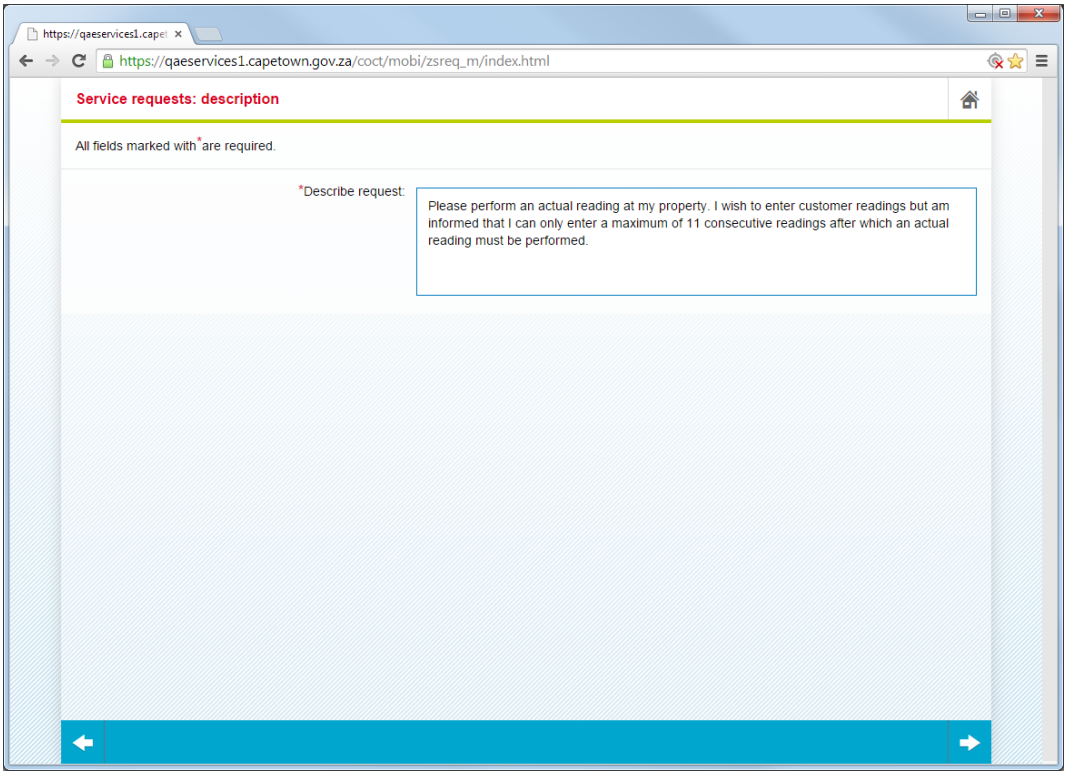

Figure 11: Account-Based Service – Description of Service

Identification of Account Number

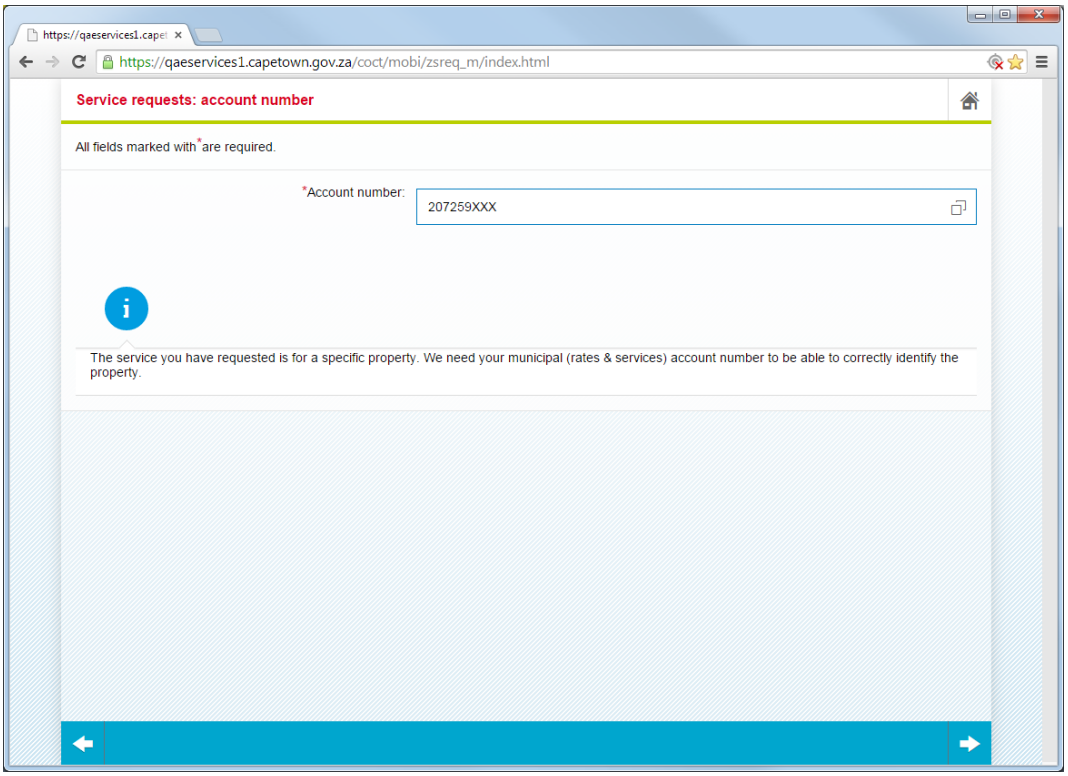

Figure 12: Account-Based Service – Specify Account Number

Confirmation of Address

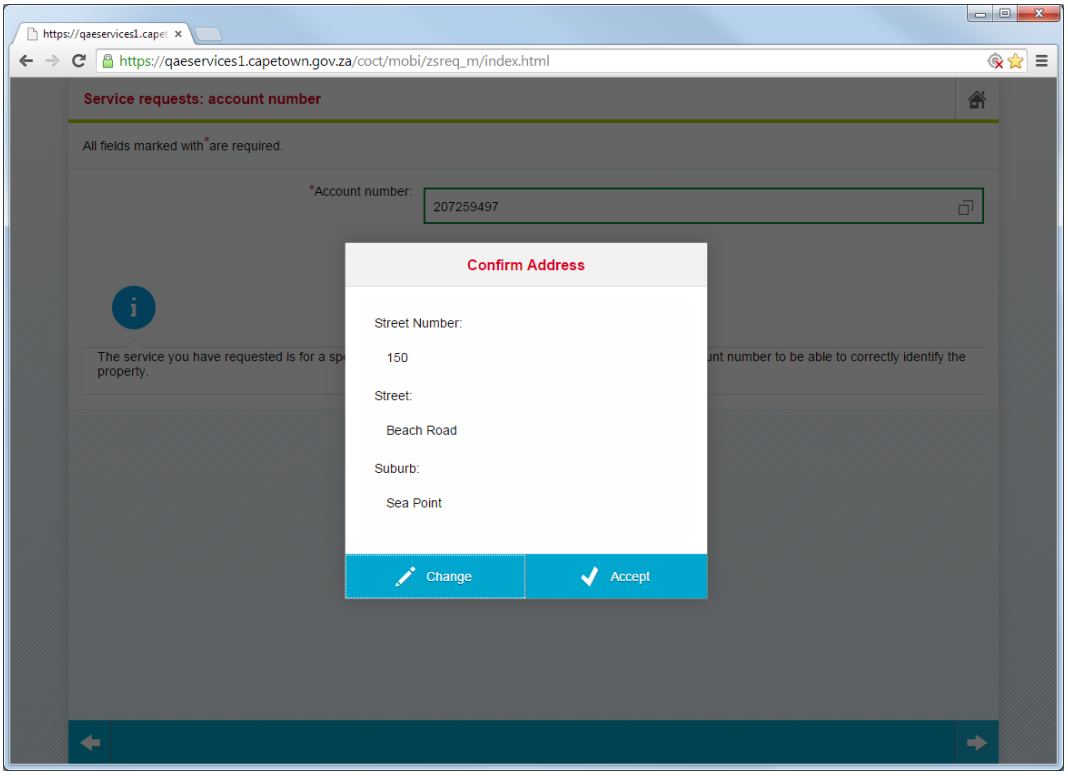

Figure 13: Account-Based Service – Confirmation of Address

Contact Details

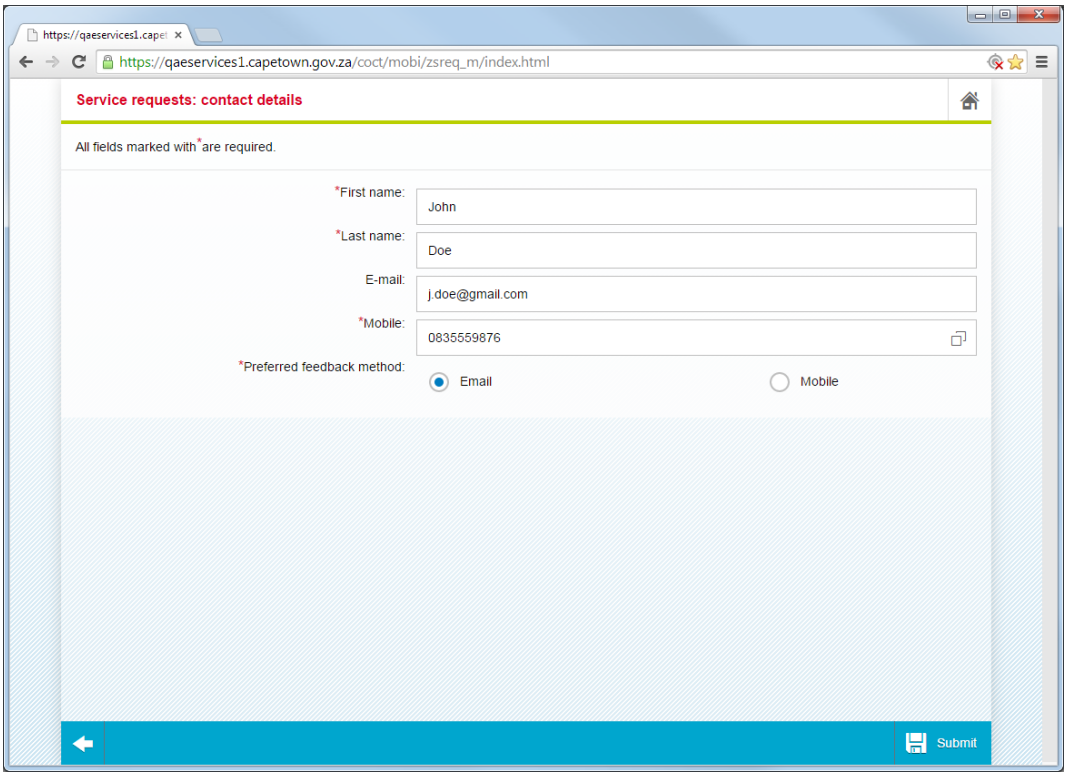

Figure 14: Account-Based Service – Contact Details

Confirmation of Service Request Created

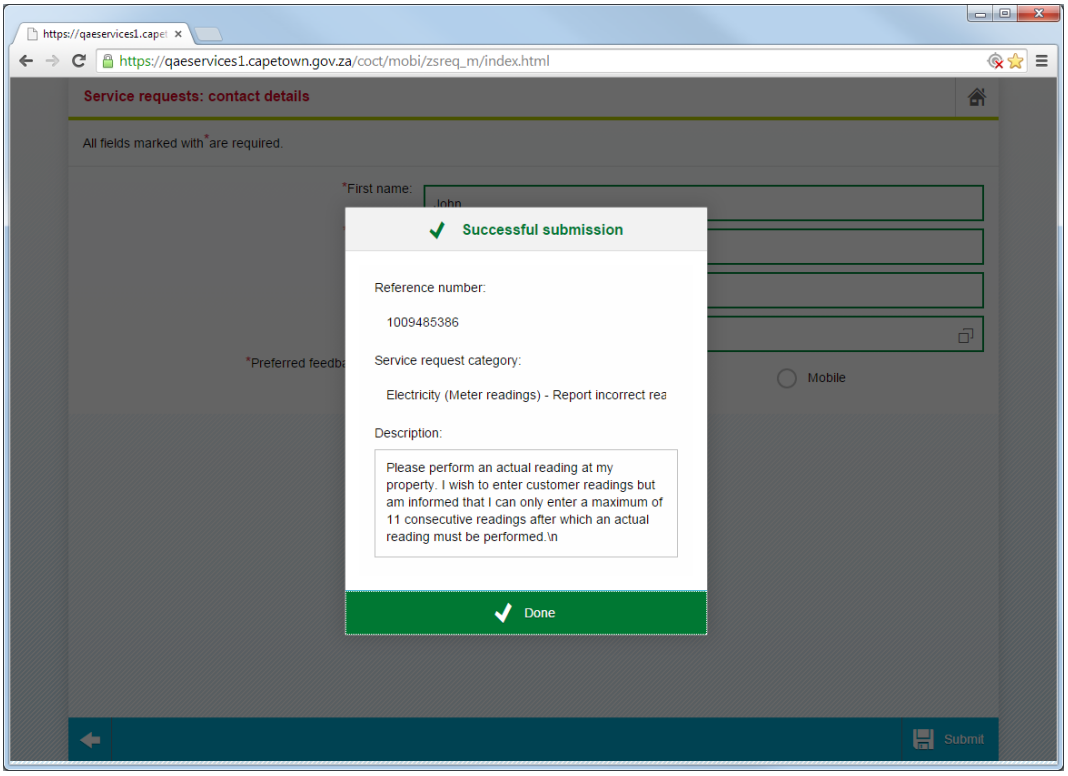

Figure 15: Account-Based Service – Confirmation of Request

A.2.3 Service Request Status Get

Provide Reference Number

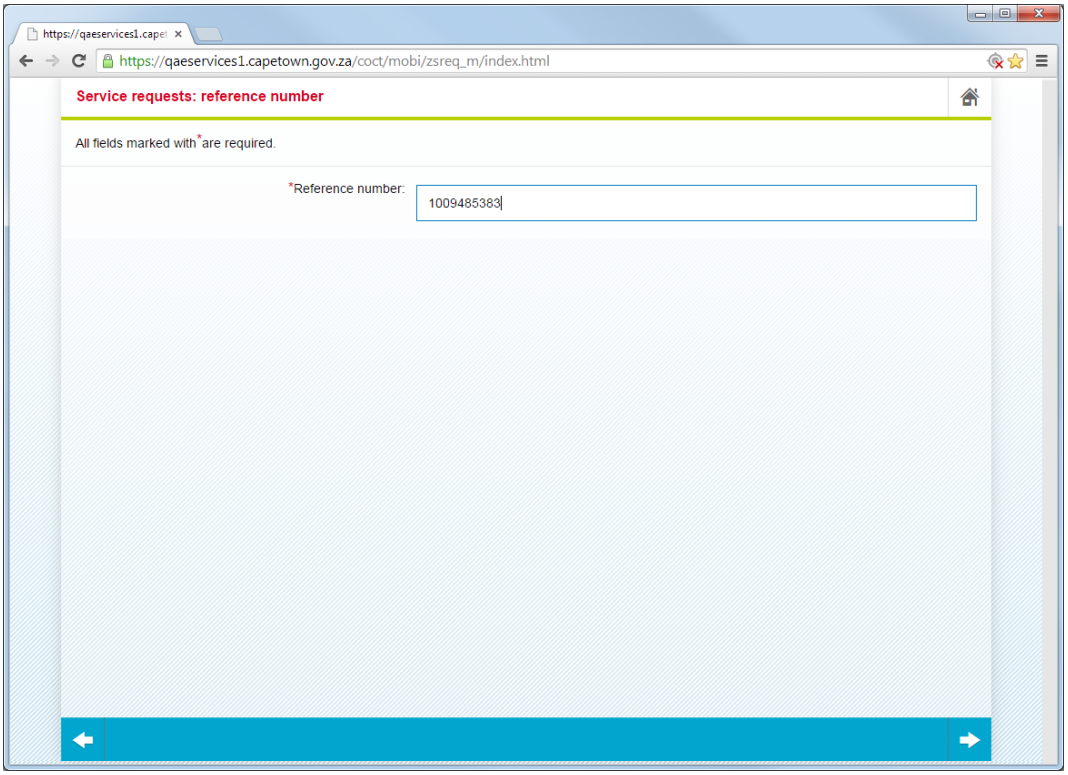

Figure 16: Service Request Status – Provide Reference Number

Status Confirmation

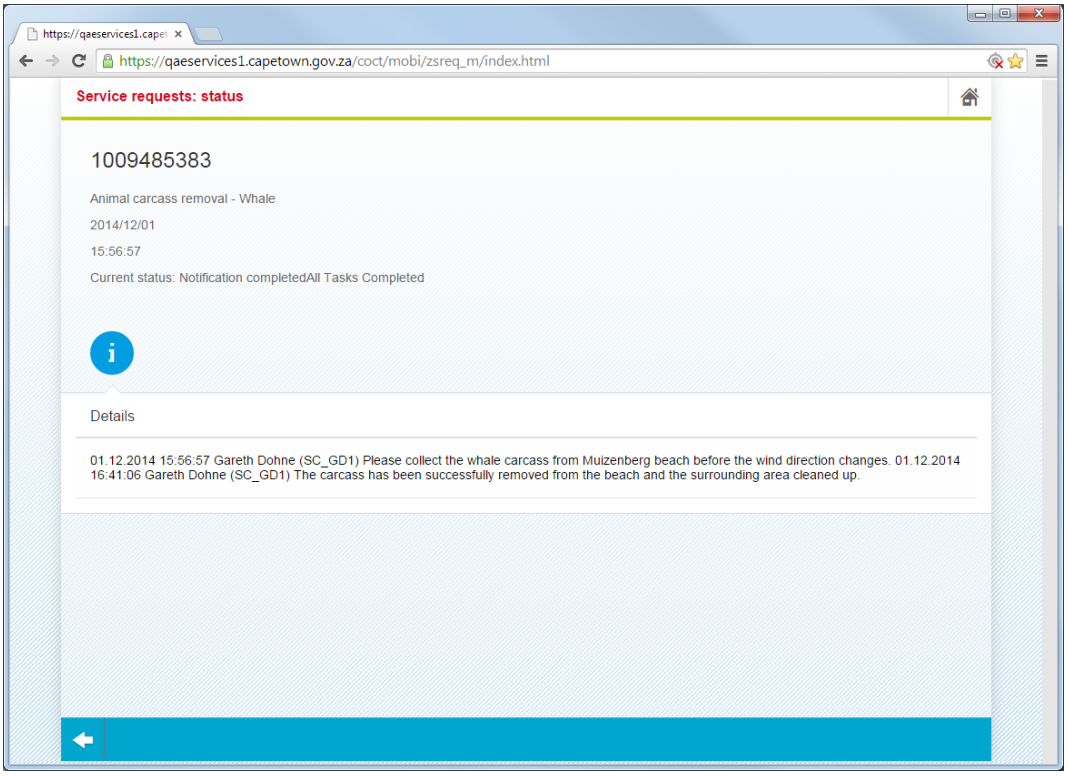

Figure 17: Service Request Status – Service Request Status Confirmation

A.2.4 My Service Requests

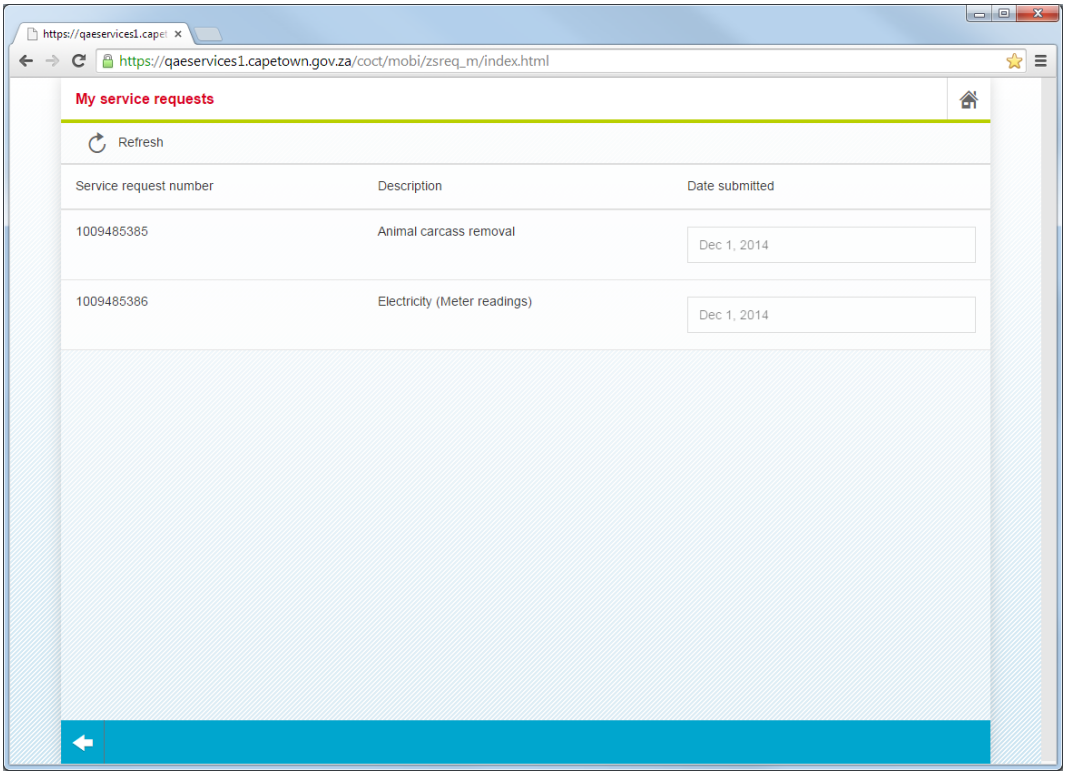

Figure 18: My Service Requests (Registered & Logged-In User)

A.3 User Registration & Maintenance DEPRECATED

A.3.1 User Registration (Create User Profile) Deprecated

Contact Details

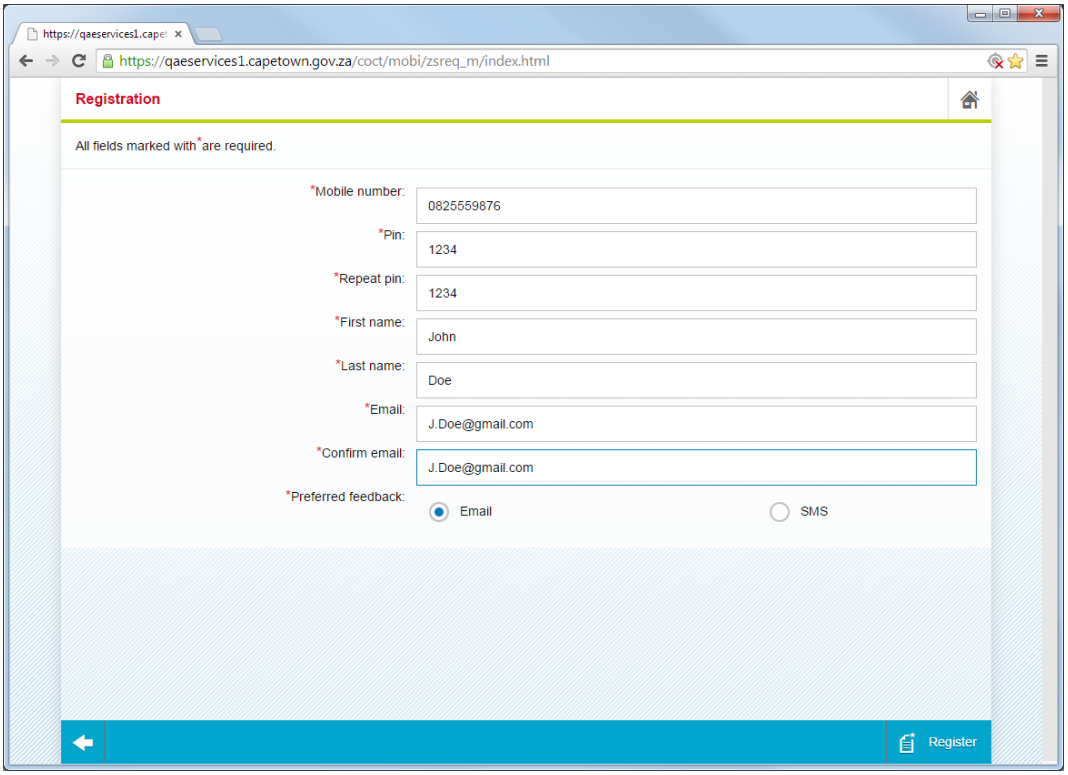

Figure 19: Registration Contact Details

A.3.2 Profile Maintenance Deprecated

Registered User Login

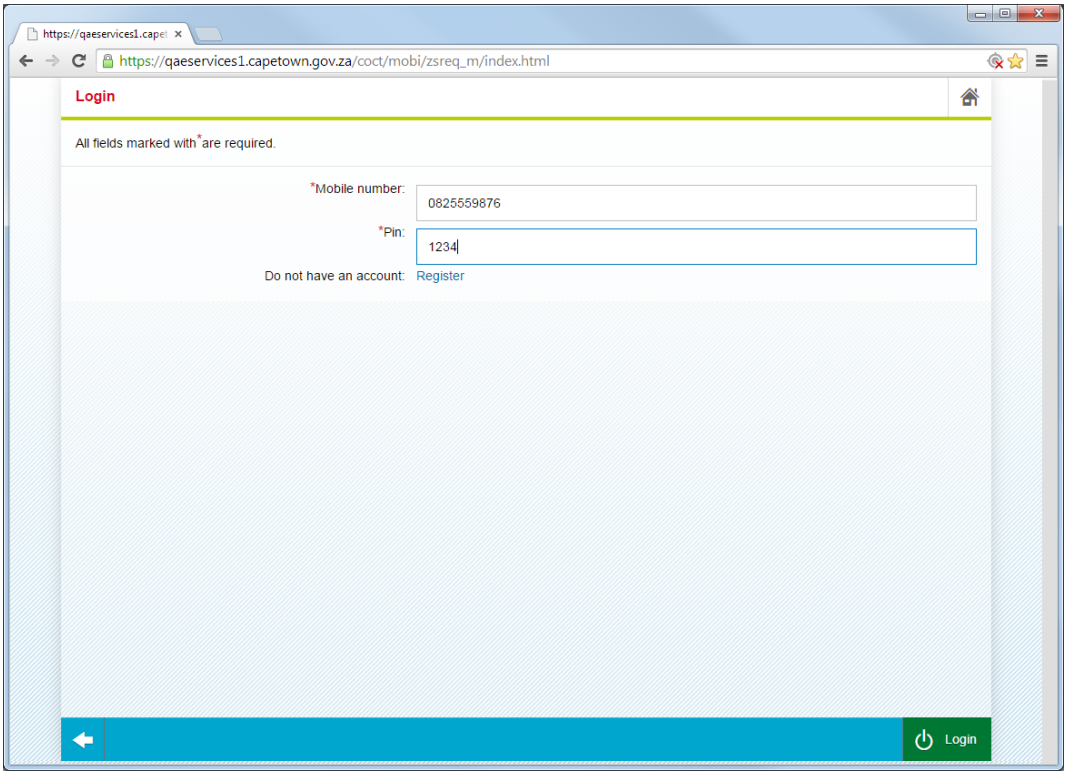

Figure 20: Profile Maintenance – Login Registered User

Name, Email Address & Feedback Preference

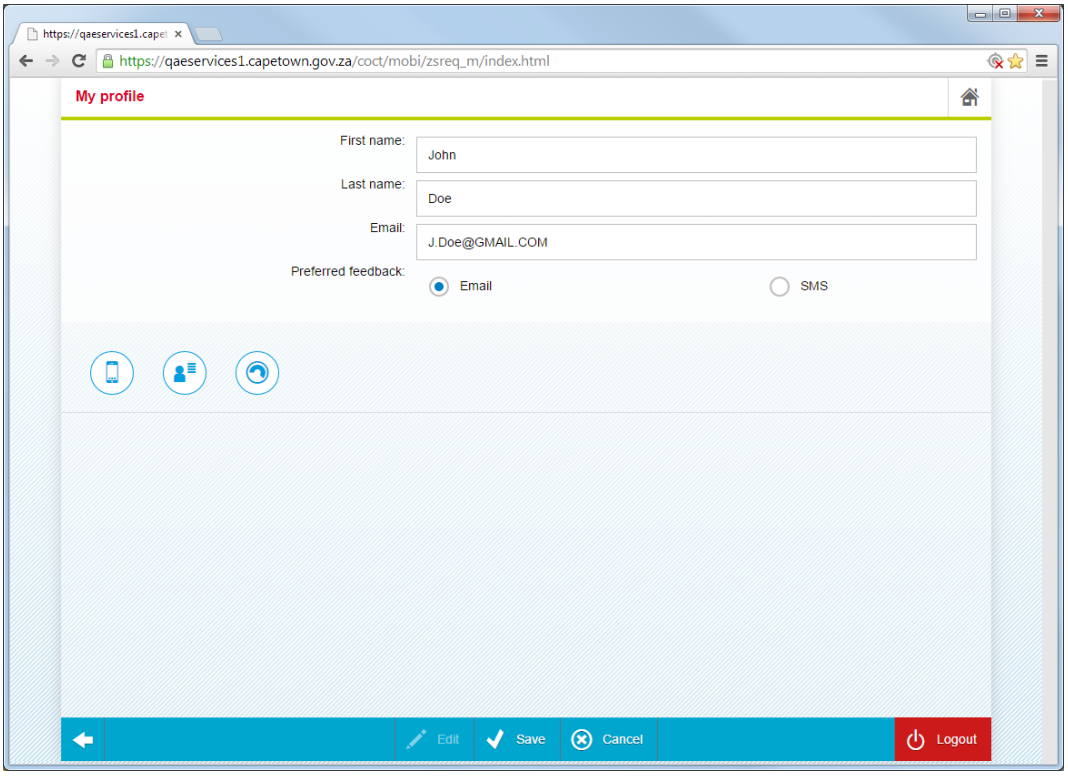

Figure 21: Profile Maintenance – Name & Email Details

Mobile Number Maintenance

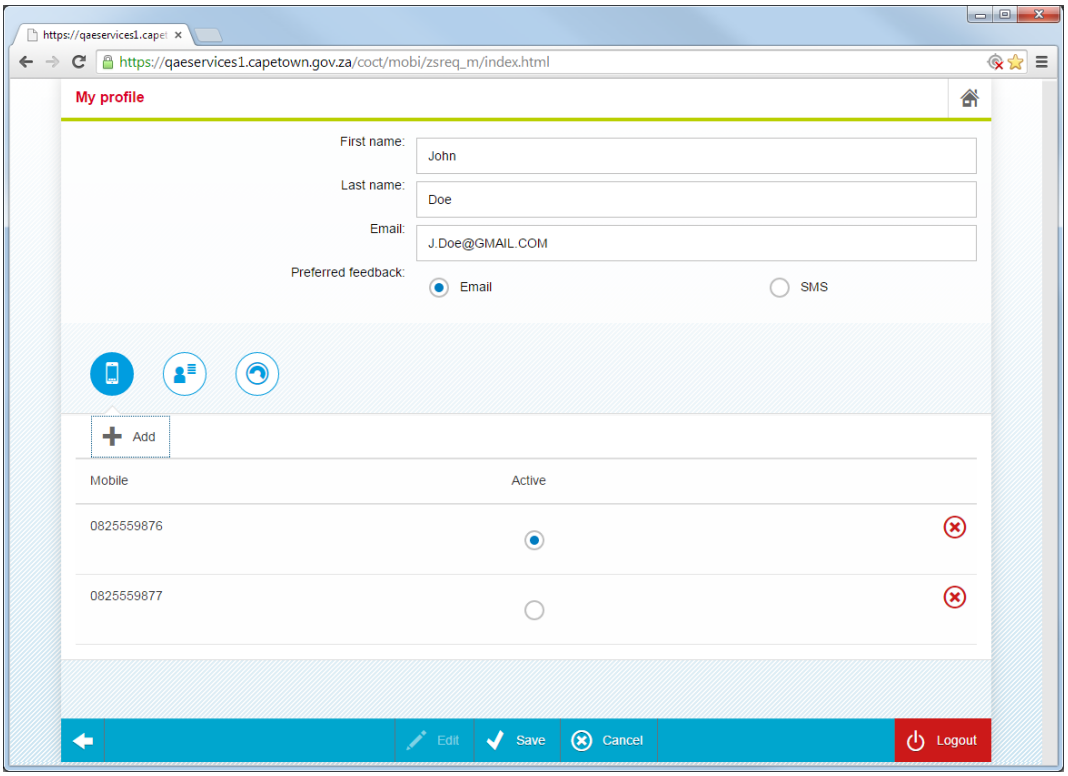

Figure 22: Profile Maintenance – Mobile Number Maintenance

Account Number Maintenance

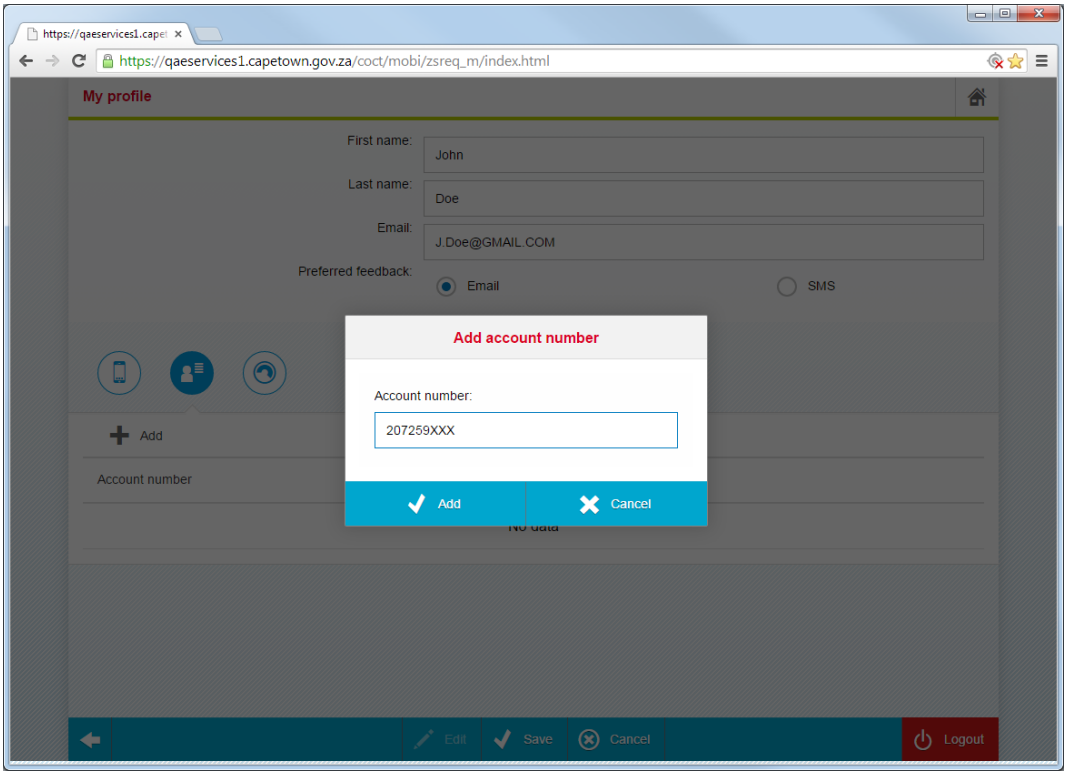

Figure 23: Profile Maintenance – Account Number Maintenance

PIN Code Maintenance

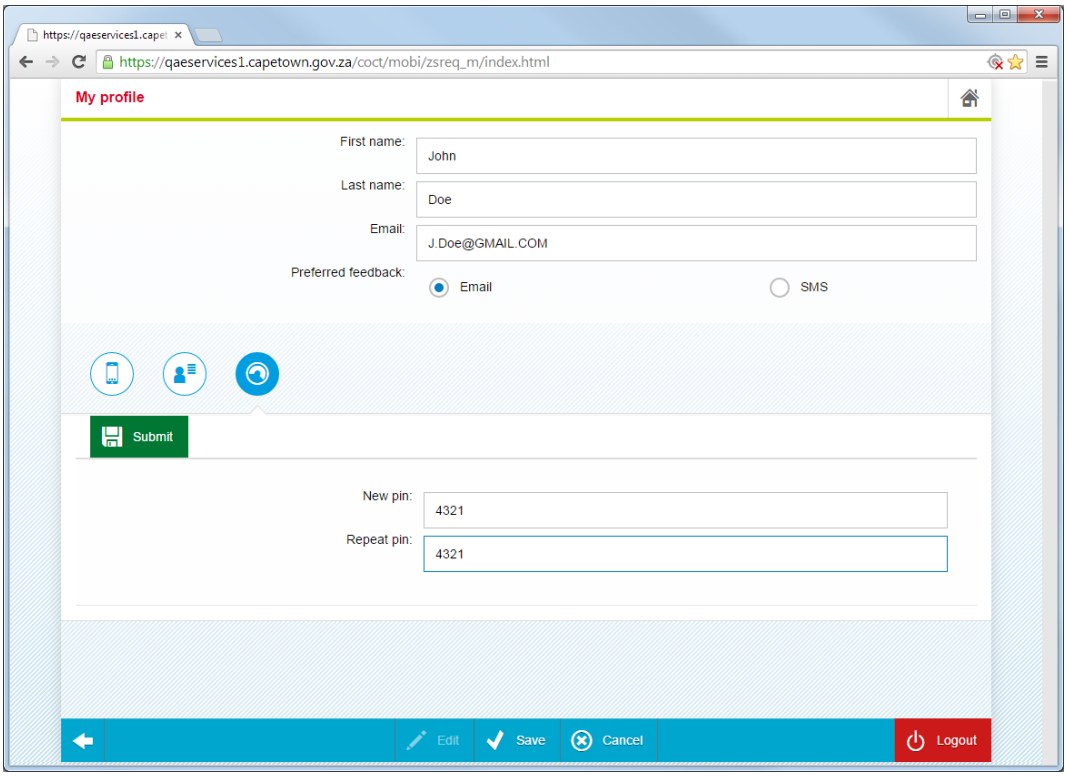

Figure 24: Profile Maintenance – PIN Code Maintenance

Appendix B: Sample Service Responses

B.1 Resource: Session

B.1.1 API-WSR-SE01: Get Session Identifier

Figure 25: Service Test – Get Session Identifier

B.2 Resource: Configuration

B.2.1 API-WSR-CN01: Get Types

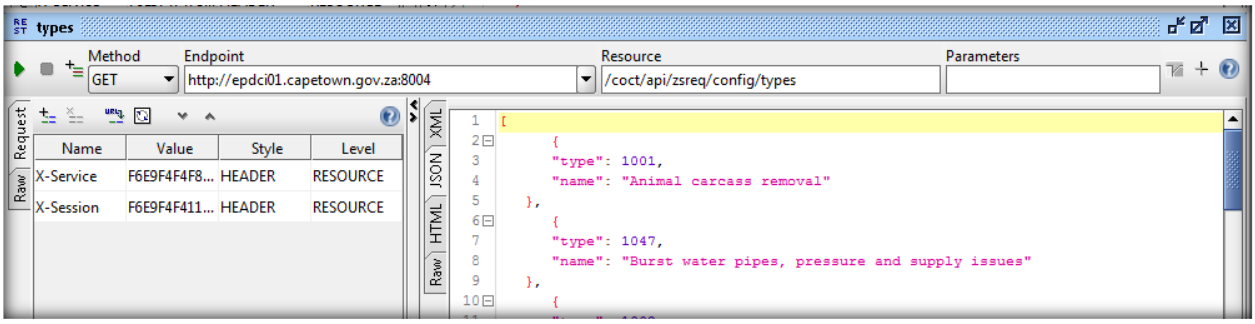

Figure 26: Service Test – API-WSR-CN01 Get Types

B.2.2 API-WSR-CN02: Get Subtypes

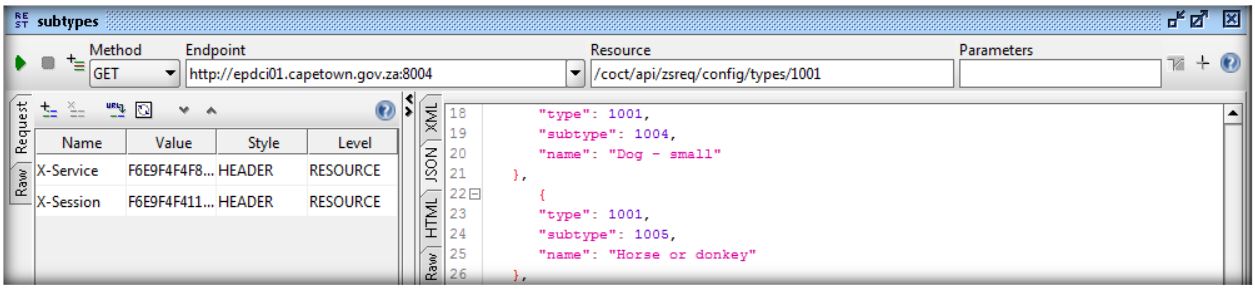

Figure 27: Service Test – API-WSR-CN02 Get Subtypes

B.2.3 API-WSR-CN03: Get All Subtypes

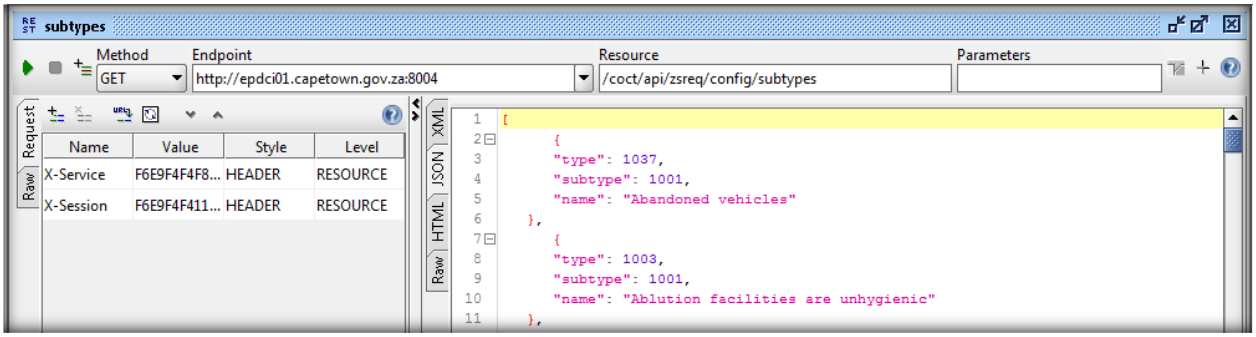

Figure 28: Service Test – API-WSR-CN03 Get All Subtypes

B.2.4 API-WSR-CN04: Get Service Configuration

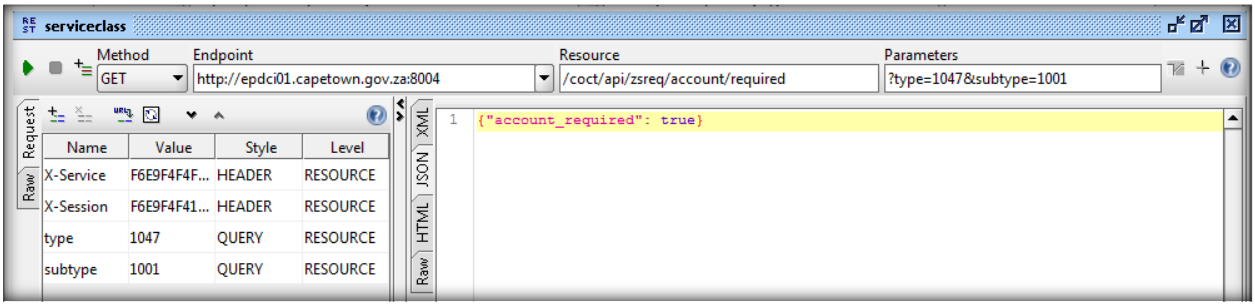

Figure 29: Service Test – API-WSR-CN04 Get Service Configuration

B.3 Resource: Account

B.3.1 API-WSR-AC01: Get Service Class

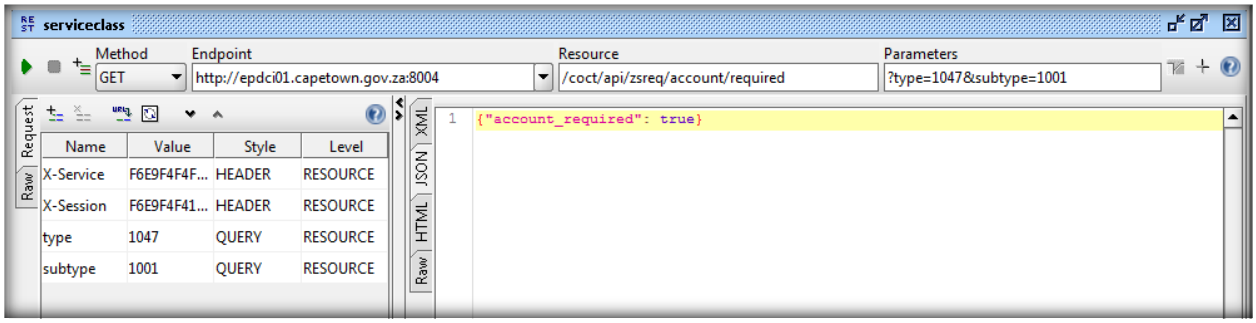

Figure 30: Service Test – API-WSR-CN04 Get Service Class

B.3.2 API-WSR-AC01: Get Account Details

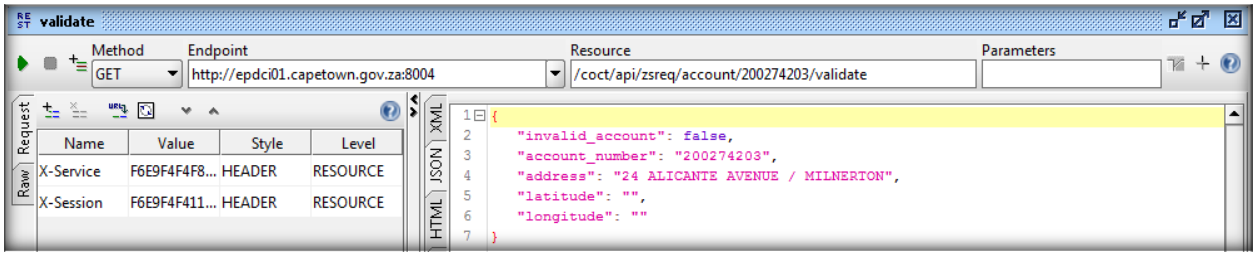

Figure 31: Service Test – API-WSR-CN04 Get Account Details

B.4 Resource: Service Request

B.4.1 API-WSR-SR01: Create Service Request

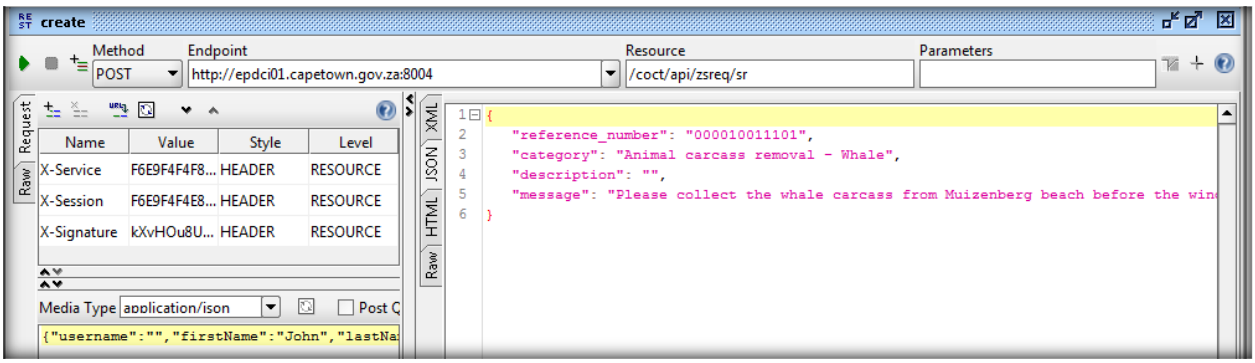

Figure 32: Service Test – API-WSR-SR01 Create Service Request

B.4.2 API-WSR-SR02: Get Service Request Detail

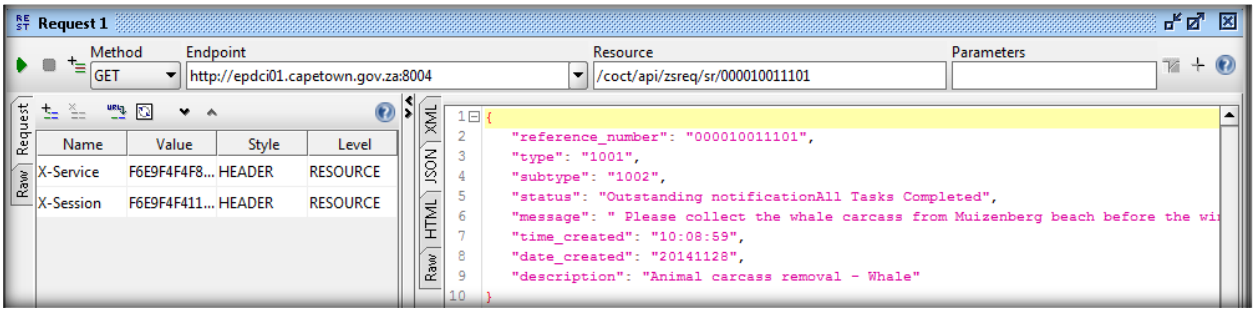

Figure 33: Service Test – API-WSR-SR02 Get Service Request Detail

B.5 Resource: User DEPRECATED

B.5.1 API-WSR-US01: Create User Profile

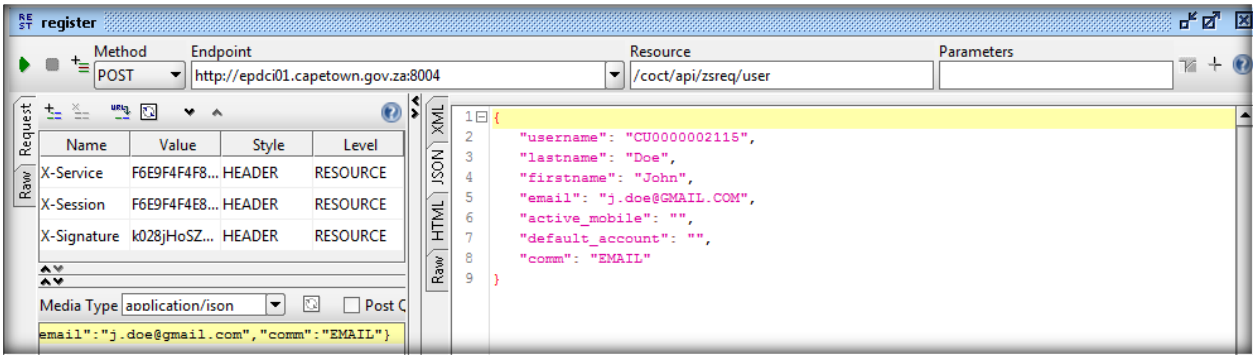

Figure 34: Service Test – API-WSR-US01 Create User Profile

B.5.2 API-WSR-US02: Get User Profile

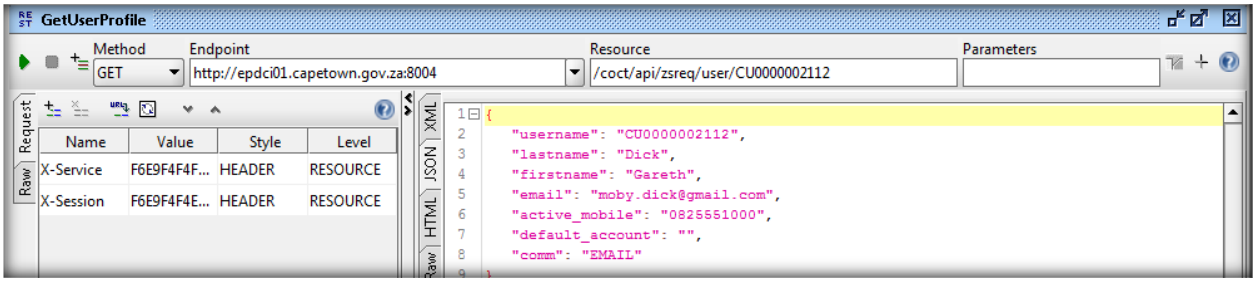

Figure 35: Service Test – API-WSR-US02 Get User Profile

B.5.3 API-WSR-US03: Change User Profile

Figure 36: Service Test – API-WSR-US03 Change User Profile

B.5.4 API-WSR-US04: Get User's Mobile Numbers

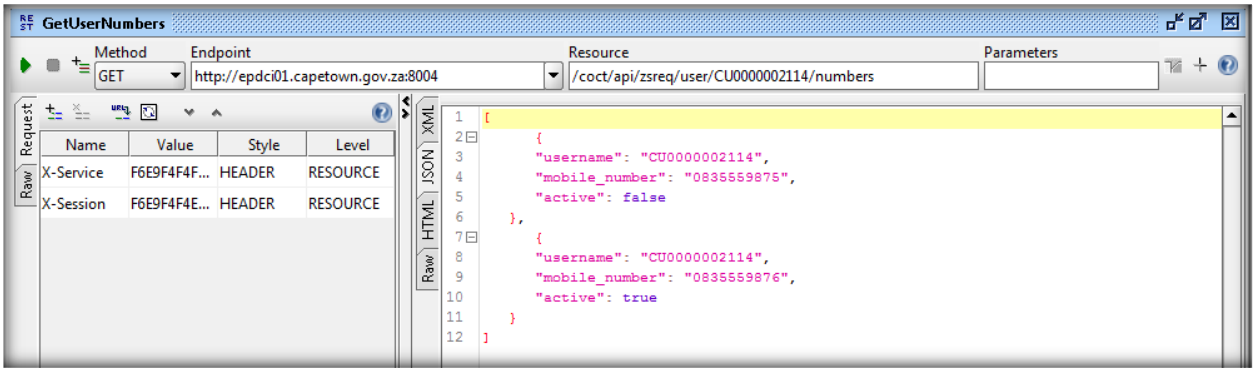

Figure 37: Service Test – API-WSR-US04 Get User's Mobile Numbers

B.5.5 API-WSR-US05: Add User's Mobile Number

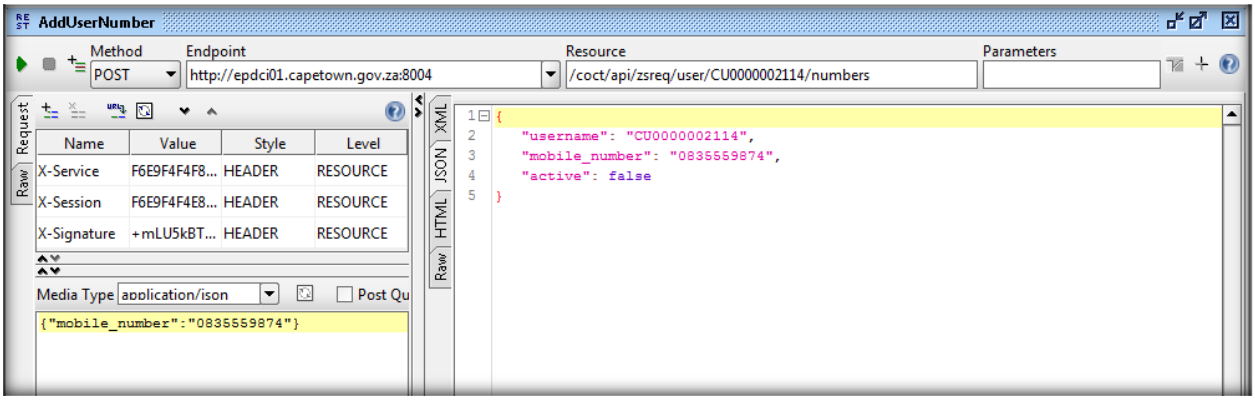

Figure 38: Service Test – API-WSR-US05 Add User's Mobile Numbers

B.5.6 API-WSR-US06: Change User's Active Mobile Number

Figure 39: Service Test – API-WSR-US06 Change User's Active Mobile Number

B.5.7 API-WSR-US07: Delete User's Mobile Number

Figure 40: Service Test – API-WSR-US07 Delete User's Mobile Numbers

B.5.8 API-WSR-US08: Get User's Account Number

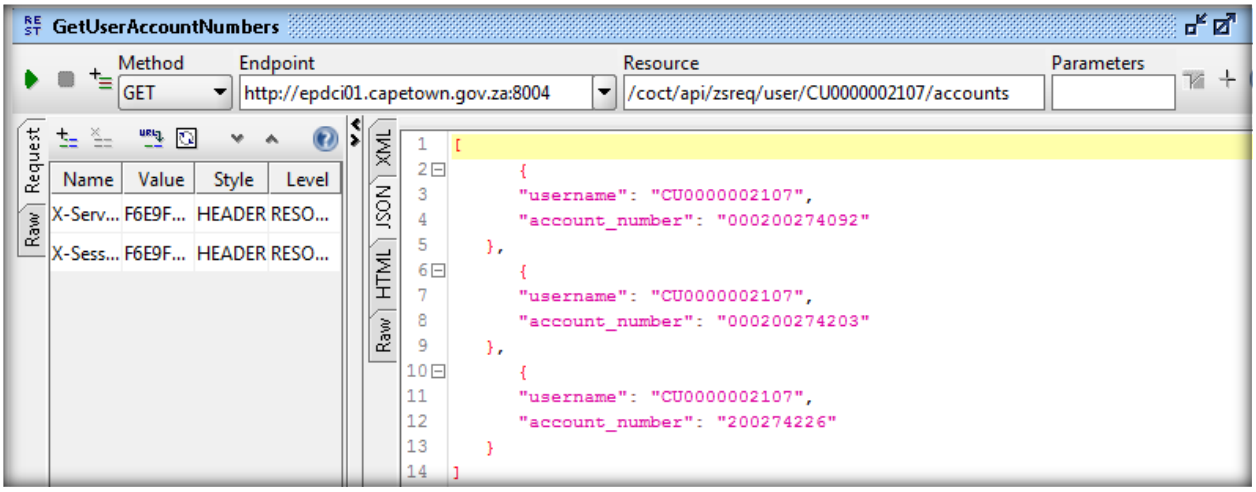

Figure 41: Service Test – API-WSR-US08 Get User's Account Numbers

B.5.9 API-WSR-US09: Add User's Account Number

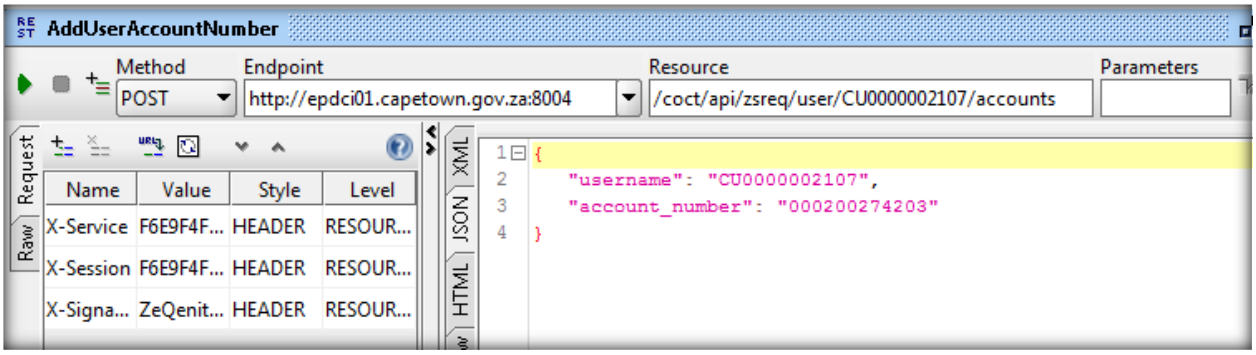

Figure 42: Service Test – API-WSR-US09 Add User's Account Number

B.5.10 API-WSR-US10: Delete User's Account Number

Figure 43: Service Test – API-WSR-US10 Delete User's Account Number

B.5.11 API-WSR-US11: Change User's Authentication PIN Code

Figure 44: Service Test – API-WSR-US11 Change User's Authentication PIN Code

B.5.12 API-WSR-US11: Authenticate User

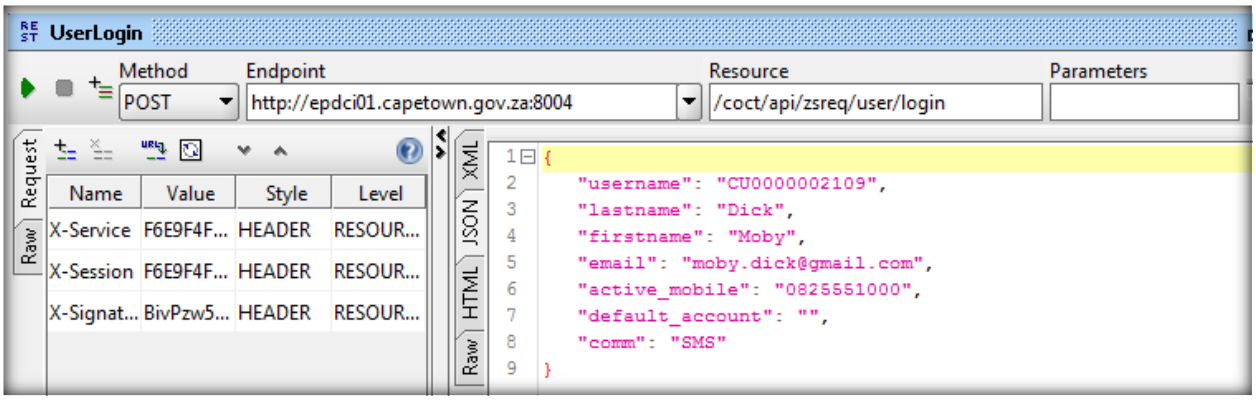

Figure 45: Service Test – API-WSR-US12 Authenticate User