civica

AUTHORITY v.6

Mapping Integration - Developer Guide

NOTICE

Revision History

Date	Version No.	Author	Description
22 February 2005	1.1	Annabel Mackay	Initial Revision
15 June 2005	6	Annabel Mackay	Added “map” option to Assets launch filter
30 June 2005	7	Annabel Mackay	Added multiple network card details
22 June 2006	8	Annabel Mackay	Added for_acc to neighbour notify where clause
26 June 2006	9	Annabel Mackay	Added for_acc to table structures table
15 April 2009	10	Steve Carter	Add lay_nme and gis_ref to aualmapl table description.
22 February 2005	1.1	Annabel Mackay	Initial Revision
01 October 2009	2.0	Tony Singh	Updated Template

1. Table of Contents

2. Introduction

The **AUTHORITY** textual information system is enabled to integrate with a spatial or graphical information system such as GIS or CAD software. This allows:

- Selected textual objects, such as properties or assets, to be plotted on a map or passed as input to spatial processes
- Selected spatial objects to have attribute details displayed in the textual system or passed as input to textual system processes

Within this integration, the design philosophy is, wherever possible, to perform textual queries within the textual system and spatial queries within the spatial system. Thus, integration is at the menu level with both software products running on the PC desktop concurrently.

Purpose

This document specifies the functions required within the mapping system to enable integration with **AUTHORITY 5.0**.

Objectives

To describe the programs required in a mapping system to:

- Receive information from **AUTHORITY** and perform actions based upon that information
- Send information to **AUTHORITY** and request that **AUTHORITY** perform actions upon that information

Audience

This document is for use by those writing mapping system programs to enable integration with **AUTHORITY**.

Prerequisites

Those writing integration programs will require skills in:

- SQL
- The mapping system's command language
- The mapping system's scripting or programming language
- Basic MS Windows programming (i.e. batch files, running commands)

No prior knowledge of **AUTHORITY** is required.

Symbols/Terminology

Mapping system	The spatial information system, graphical information system, GIS or CAD system
AUTHORITY Link	A collective name for the software in the mapping system that enables integration with **AUTHORITY**
	This symbol indicates functions that must be written

by the **AUTHORITY Link** developer for the mapping system.

This symbol indicates tip to make integration functions more user friendly.

Programming Considerations

The **AUTHORITY Link** software may be provided as a single program or as several programs that call each other as required.

The programs can be written using any language of the **AUTHORITY Link** developer's choice.

Functions requiring user interaction can be implemented using any method of the developer's choice. For example, as tool bar icons, drop down menu items, etc. A separate widget can be provided for each function, or a single "Authority" widget could be provided, selection of which prompts the user nominate the required function from a pick list.

AUTHORITY does not need to be installed to develop the **AUTHORITY Link** software. Most functions are based on data exchange via database tables and so access to an RDBMS is recommended. Other functions call **AUTHORITY** programs directly; this document provides command syntaxes for such functions.

3. Logon and Start Up

Integration between **AUTHORITY** and a mapping system requires both systems to be running on the PC desktop at the same time.

> Whilst both systems must eventually be running, only one system need be running to begin; the first time an integration function is used in one system, it should start up the other system.

3.1 Starting Integration from AUTHORITY

3.1.1 Setup Mapping Link Program (setup_customer.bat)

This program is used to start the mapping system, initialise it for integration with **AUTHORITY** and tell it to process requests from **AUTHORITY**.

By default **AUTHORITY** looks for a file:

- Called **Setup_customer.bat**
- Installed on the client PC
- Installed in a location that is included on the Windows PATH (e.g. /Windows), or the Windows PATH be amended to include the file's location

However, **AUTHORITY** can be configured to look for a file with a different name available on the network.

This program or programs it calls, should:

1. Check if the mapping system is running on the PC
2. If not, start the mapping system.
3. Determine the PC identifier.

   See *PC Identifier*.
4. Check whether the mapping software has an active connection to the **AUTHORITY** database.
5. If not, establish a connection.

   Most often mapping systems use ODBC to connect to the **AUTHORITY** database using an appropriate ODBC driver for the RDBMS on which **AUTHORITY** runs (e.g. Informix or Oracle).

   

   Allow the user to configure the default database connection and change it upon logon.
6. Select rows from **AUALMAPL** for processing:

```
select
  *
from
  aualmapl
where
  ip_adr = '[PC identifier]'
  and dir_flg = 'M'
```

7. If the Function Type field, **fnc_typ**, of the rows selected is **1** or **2**, initiate the **Display Parcels On Map Program**.
8. Delete rows from **AUALMAPL** that have been processed:

```
delete from aualmapl
```

```
where
   ip_adr = '[PC identifier]'
   and dir_flg = 'M'
```

3.2 Starting Integration from the Mapping System

3.2.1 Setup AUTHORITY Link Program

This program is used to start **AUTHORITY**, and initialise the mapping system for integration with **AUTHORITY**.

This program can be executed either, once when the mapping system starts up, or each time an integration function is used. This program should:

1. Check if **AUTHORITY** is running on the PC

   For example, on Windows NT,

   ```
   query process | find "authoritymenu.exe"
   if not errorlevel 1 goto startauthority
   :startauthority
   Sys C:\Program Files\Authority 46 Menu\AuthorityMenu.exe
   ```

2. If not, start **AUTHORITY**.

   AUTHORITY is started using the Windows program **authoritymenu.exe**. **Authoritymenu.exe** is installed on the PC as part of the **AUTHORITY** client software installation.
   To start **authoritymenu.exe** from the Windows command prompt or within a batch file, its' location must be included on the PCs **PATH** environment variable, or when called, the full path to the program must be used (e.g. **C:\Program Files\Authority 46 Menu\AuthorityMenu.exe**).

3. Determine the PC identifier.

   See *PC Identifier*.

4. Check whether the mapping software has an active connection to the **AUTHORITY** database.

5. If not, establish a connection.

   Most often mapping systems use ODBC to connect to the **AUTHORITY** database using an appropriate ODBC driver for the RDBMS on which **AUTHORITY** runs (e.g. Informix or Oracle).

   

   Allow the user to configure the default database connection and change it upon logon.

4. Display Functions

4.1 Display Properties on Map

4.1.1 Display Parcels On Map Program

This program is used to display parcels on a map that have been sent from **AUTHORITY**, and if requested initiate the Neighbour Notification program.

This program is initiated by the mapping system after it selects rows from the **AUTHORITY** data exchange table, **AUALMAPL**, to process.

This program should:

1. Determine the map layer on which to display parcels.

   

   Allow the user to configure the default map layer and field name containing the **AUTHORITY** parcel number.

   Optionally, the map layer could be set using the **AUTHORITY** module type (**mdu_ref**) listed in **AUALMAPL**.
   For example,

```
if mdu_ref = 'PR'
      let layer = cadastre
if mdu_ref = 'AS'
      let layer = assets
```

This function could hard code each module type to a layer name, or it could reference a configuration file or database table that is used to match module type to layer name. The latter option would enable users to redefine the layers as required.

2. Optionally, match parcel numbers (**pcl_num**) listed in **AUALMAPL** to mapping system parcel identifiers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.

3. Optionally, check if parcel exists on map.

   If not, add to list of missing parcels.

4. Build a list of subject parcels (**pcl_typ** = 1).
5. Build a list of neighbour parcels (**pcl_typ** = 2).
6. Plot/shade **subject** parcels in one colour/symbology.
7. Plot/shade **neighbour** parcels in a different colour/symbology.
8. Display message listing missing parcel identifiers.
9. If **fnc_typ** equals **2**, initiate the **Neighbour Notification Program**.

4.2 Display Properties in AUTHORITY

This function is used to display parcels in **AUTHORITY** that have been selected on a map. Either one of two methods can be used to do this.

AUALMAPL Method

The AUALMAPL method is equivalent to the way **AUTHORITY** displays parcel on a map but operates in reverse.
The mapping system inserts rows into the **AUTHORITY** data exchange table, **AUALMAPL**, for each parcel to be displayed in **AUTHORITY**, and then executes an **AUTHORITY** API command to process the requests.
Parcels are displayed in the **AUTHORITY Customer Services Property Enquiry** (**e_pr005**) program. This program displays **AUTHORITY Property** data and shows links to any other **AUTHORITY** module for which the parcel is involved.

Ulaunch Method

The Ulaunch method executes an **AUTHORITY** API command to display parcels in any one of many **AUTHORITY** programs. For example, an API command is executed to display animal records linked to a parcel in the **Animal Enquiry** program.

Differences

Feature	Method	
	AUALMAPL	**Ulaunch**
Number of widgets required in mapping system	One	Many One for each **AUTHORITY** program in which data is to be displayed.
Result in **AUTHORITY**	Always displays a result	May not display a result Only displays a result if parcel has links to entities in nominated program. For example, if the **Animal Enquiry** program is launched a record will only be displayed if the parcel has linked **AUTHORITY** Animal records.

4.2.1

4.2.2 Display Parcels in AUTHORITY Program (AUALMAPL Method)

This program should:

1. Run the **Setup AUTHORITY Link Program** if it is not executed as part of the mapping system start up.
2. Prompt the user to select parcels of interest.
   Alternately, the last parcel selection may be used.
3. Optionally, match mapping system parcel identifiers to **AUTHORITY** parcel numbers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.
4. Insert a row into **AUALMAPL** for each selected parcel.
   For example,

```
insert into aualmapl values
      ('[PC identifier]',
      'A',
      1,
      1,
      'PR',
      '[Parcel Number]',
      [Parcel Number],
      1,
      null)
```

5. Execute the **AUTHORITY** API command to run the program **i_al033**.
   For example,

```
ulaunch /f GISREQ
```

   
   Allow the user to configure the ulaunch command.

   See *Running Authority Programs* for information on running **i_al033**.

4.2.3 Display Parcels in AUTHORITY Program (Ulaunch Method)

This program should:

1. Run the **Setup AUTHORITY Link Program** if it is not executed as part of the mapping system start up.
2. Prompt the user to select parcels of interest.
   Alternately, the last parcel selection may be used.
3. Optionally, match mapping system parcel identifiers to **AUTHORITY** parcel numbers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.
4. Execute the **AUTHORITY** API command to run the program in which records matching the selected parcels are to be displayed.
   For example, to displays the selected parcels in the **AUTHORITY** Rates Enquiry program:

   ```
   ulaunch /f RTENQ -f pcl_num [parcel numbers]
   ```

   
   Allow the user to configure the ulaunch command.

   See *Running Authority Programs* for information on running **AUTHORITY** programs.

5. Neighbour Notifications

The combination of functions in **AUTHORITY** and the mapping system enable Neighbour Notifications to be processed.
The neighbour notification process involves 2 functions:

1. Neighbour Parcel Linking
   In **AUTHORITY,** links can be created between applications and properties to show:
   - The properties for which an application is being submitted (i.e. **include** parcels)
   - The properties that are affected by an application (i.e. **neighbour** parcels).

   Typically, application-to-property links are entered individually using the **AUTHORITY Property Links** program. This process may involve consultation of hardcopy maps to identify affected properties, and where an application affects many properties this process may be time consuming.
   Mapping system software can be used to more easily identify affected properties, and after selecting affected properties in the mapping system, they can be sent to **AUTHORITY** to be linked to an application using an automated process.
2. Correspondence Production
   AUTHORITY includes functionality to create correspondence about **AUTHORITY Applications**. When correspondence documents created they are stored on Council's computer system and **AUTHORITY Desktop** document links are created between the **AUTHORITY Application** that the correspondence is about and the document file.
   The **Application** correspondence production process is initiated within **AUTHORITY** or MS Word for individual document requests, but where the same document is requested for multiple recipients, as in Neighbour Notifications, the correspondence production process is initiated from a mapping system.

5.1 Neighbour Notifications Program

This program determines what aspects of the neighbour notification process are to be completed, selects neighbour parcels, inserts rows into the **AUTHORITY** data exchange table **AUALMAPL** for neighbour notification requests, and then executes an **AUTHORITY** API command to process the requests.
This program should:

1. Ask the user whether they wish to create Applicant correspondence. By default this should be set to Yes.

2. If the user choses to create Applicant correspondence, ask the user to select which **AUTHORITY** External Document Type(s) should be created for the Applicant.
   The following SQL statement lists available document types:

```
Select
      ext_typ,
      ext_dsc
from
      audmextp
where
      mdu_ref = 'DD'
      and ( for_doc is null
      or for_doc = [Application type] )
```

   Where [Application type] First 3 digits of AUALMAPL:FOR_ACC with zero padding removed.
   For example, if fmt_acc is 021.2006.00037451.001, then [Application type] is 21.

3. Ask the user whether they wish to create Neighbour Parcel links.
   By default this should be set to Yes.
4. Ask the user whether they wish to create Neighbour Parcel Owner/Occupier correspondence.
   By default this should be set to Yes.

5. If the user choses to create Neighbour Parcel Owner/Occupier correspondence, ask the user to select which **AUTHORITY** External Document Type(s) should be created for the neighbour parcels.
   The following SQL statement lists available document types:

```
select
      ext_typ,
      ext_dsc
from
      audmextp
where
      mdu_ref = 'DD'
      and ( for_doc is null
      or for_doc = [Application type] )
```

   Where [Application type] First 3 digits of AUALMAPL:FOR_ACC with zero padding removed.
   For example, if fmt_acc is 021.2006.00037451.001, then [Application type] is 21.

Functions 1-5 can be presented in any of the following ways:

- Questions that follow one another
- A dialog box with toggles for each option and a drop down list of letter types which becomes un-ghosted if the user chooses to create letters
- A system parameter that always applies for neighbour notifications
- A system parameter that sets defaults for neighbour notifications but can be overridden each time notifications are processed

6. If the user choses to create Neighbour Parcel links or create Neighbour Parcel Owner/Occupier correspondence select neighbour parcels.
   Select, or prompt the user to select, neighbour and other parcels affected by an application.
   The affected parcel selection process may be implemented so as to prompt the user to select parcels using any selection method they chose, or it may be an automated process that uses the mapping system's adjacent parcel selection command and then prompts the user for additional ad-hoc parcel selection.

7. If neighbour parcels are selected, optionally, match mapping system parcel identifiers to **AUTHORITY** parcel numbers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.

8. If the user choses to create Applicant correspondence insert a row into **AUALMAPL** for each subject parcel.
   For example,

```
insert into aualmapl values
      ('[PC identifier]',
      'A',
      1,
      3,
      'DD',
      '[Application number from Authority]',
      [Subject parcel number from Authority],
      1,
      '[External Document Type]')
```

9. If the user choses to create Neighbour Parcel links insert a row into **AUALMAPL** for each neighbour parcel.
   For example,

```
insert into aualmapl values
      ('[PC identifier]',
      'A',
      2,
      4,
      'DD',
      '[Application number from Authority]',
      [Neighbour Parcel Number],
      1,
      null)
```

10. If the user choses to create Neighbour Parcel Owner/Occupier correspondence insert a row into **AUALMAPL** for each neighbour parcel.
    For example,

```
insert into aualmapl values
      ('[PC identifier]',
      'A',
      2,
      3,
      'DD',
      '[Application number from Authority]',
      [Neighbour Parcel Number],
      1,
      '[External Document Type]')
```

11. Execute the **AUTHORITY** API command to run the program **i_al033**.
    For example,

```
ulaunch /f GISREQ
```

Allow the user to configure the ulaunch command.

See *Running Authority Programs* for further information about running **i_al033**.

I_al033 will read the table **AUALMAPL** and create:

- Letter requests for the Applicant
- Property links between the **Application** and affected **Properties** of type **Neighbour**
- Letter requests for neighbour **Property** Owners and Occupiers

6. Bulk Property Correspondence

AUTHORITY includes functionality to create correspondence about **AUTHORITY Properties**. When correspondence documents created they are stored on Council's computer system and **AUTHORITY Desktop** document links are created between the **AUTHORITY Property** that the correspondence is about and the document file.
The **Property** correspondence production process is initiated within **AUTHORITY** or MS Word for individual document requests, but where the same document is requested for multiple recipients the correspondence production process is initiated from a mapping system.

6.1 Bulk Property Correspondence Program

This program selects parcels, inserts rows into the **AUTHORITY** data exchange table, **AUALMAPL**, for correspondence requests, and then executes an **AUTHORITY** API command to create the requests.
This program should:

1. Run the **Setup AUTHORITY Link Program** if it is not executed as part of the mapping system start up.
2. Prompt the user to select parcels of interest.
   Alternately, the last parcel selection may be used.
3. Optionally, match mapping system parcel identifiers to **AUTHORITY** parcel numbers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.
4. Ask the user to select which **AUTHORITY** External Document Type(s) should be created for the selected parcels.
   The following SQL statement lists available document types:

```
select
   ext_typ,
   ext_dsc
from
   audmextp
where
   mdu_ref = 'PR'
```

6. Insert a row into **AUALMAPL** for each selected parcel.
   For example,

```
insert into aualmapl values
      ('[PC identifier]',
      'A',
      1,
      3,
      'PR',
      '[Parcel Number]',
      [Parcel Number],
      1,
      '[External Document Type]')
```

5. Execute the **AUTHORITY** API command to run the program **i_al033**.
   For example,

```
ulaunch /f GISREQ
```

Allow the user to configure the ulaunch command.

See *Running Authority Programs* for further information about running **i_al033**.

I_al033 will read the table **AUALMAPL** and create letter requests for **Property** Owners and Occupiers.

7. Property Bulk Update

AUTHORITY includes functionality to modify the same data for **AUTHORITY** property records in a bulk process. This is done using **AUTHORITY Property Bulk Update** (**i_pr079**) program.
Properties can be selected for bulk updated using a variety of methods including street name, land title, and attribute type. In addition to this a mapping system can be used to identify properties for update. After selecting the properties in the mapping system, the properties can be sent to **AUTHORITY** for input to the **Property Bulk Update** program.
Selection of properties for update in a mapping system could be useful in the following situations:

- Adding a flooding attribute to all properties within 100 metres of a river to indicate flood zone: a mapping system's buffer selection tool could be used to easily identify properties within this distance of a river.
- Adding a zoning attribute to all properties of a specific size to indicate building restrictions: a mapping system's size range selection tool could be used to easily identify properties between a minimum and maximum size.

7.1 Property Bulk Update Program

This program selects parcels, writes a list of selected parcels to a file, and then executes an **AUTHORITY** API command to launch the **AUTHORITY Property Bulk Update**.
This program should:

1. Run the **Setup AUTHORITY Link Program** if it is not executed as part of the mapping system start up.
2. Prompt the user to select parcels of interest.
   Alternately, the last parcel selection may be used.
3. Optionally, match mapping system parcel identifiers to **AUTHORITY** parcel numbers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.
4. Write the list of selected parcel numbers to a text file. Each parcel number should be entered on a new line.
   For example,

   ```
   12345
   769778
   596
   ```

   The file can be saved in any directory on the file system; however, it is preferable to save onto the **AUTHORITY** server. When **AUTHORITY** reads the file it will look for it in the directory specified by the **AUTHORITY Mapping Parameters** program (**i_pr069**), **Bulk Update Import Path** field.
   The following SQL statement lists the **Bulk Update Import Path**:

   ```
   select
         chr_no1||chr_no2||chr_no3||chr_no4
   from
         aualparm
   where
         mdu_ref = 'PR'
   ```

   This path may be able to be used by the mapping system to save import files. However, this path is entered in terms of the **AUTHORITY** server and so it may not be recognised by the mapping system. For example, where the **AUTHORITY** server is a Unix server the path could be:

   ```
   /opt/apps/auth/import_files/pr/i_pr079
   ```

   If the mapping system saves files using a PC file naming convention then it will not be possible to extract this path from the **AUTHORITY** database and use it.

Allow the user to configure the default save path and file name of the import file.

5. Execute the **AUTHORITY** API command to run the program **i_pr079**. For example,

```
ulaunch /f PRBULK
```

Allow the user to configure the ulaunch command.

See *Running Authority Programs* for further information about running **i_al033**.

The **AUTHORITY Property Bulk Update** program (**i_pr079**) will be launched. The user can then use the **Load From File** button in that program to select the parcels listed in the text file.

8. Database Query Functions

The mapping system may include a range of **AUTHORITY** database query functions. Database query functions are run in the mapping system and do not need **AUTHORITY** to be running at the same time. Database query functions are SQL queries to the **AUTHORITY** database. Commonly, SQL queries are run using a tool like **dbaccess**, **SQLPlus**, or **Crystal Reports**, the results of which are displayed in a textual, tabular format. When SQL queries are run from a mapping system the results of queries can be displayed graphically on a map, textually on a map, or textually in a designated window in the mapping system.

The **AUTHORITY Link** developer may determine which database query functions will be provided. For example,

- A standard set of database query functions for use by all councils using the developer's **AUTHORITY Link** module
- Council specific database query functions created for each individual council at the time of implementation
- A function that enables the user to define their own database query functions

Allow the user to modify, copy or add ad-hoc database queries.

Thematic Mapping

Thematic mapping shades map features based on the value of attribute for each feature.
The mapping system can provide functions to produce thematic maps based on data in the **AUTHORITY** database. For example, maps showing the location of all parcels with registered dogs, heritage-listed buildings, or flood affectations.

A thematic map function may retrieve fixed data (e.g. display all parcels with animal registrations) or, may prompt the user for selection criteria (e.g. display only parcels with animals registered in the year entered by the user).

Map Labelling

Map labelling is used to print text information on map features.
The mapping system can provide functions to label maps with textual data from the **AUTHORITY** database. For example, to print owner, land title and zoning details for a selected parcel in the mapping system.
A map labelling function may display the label on the map near the object of interest or in a section of the mapping interface alongside the map.

8.1 Database Query Function Programs

8.1.1 Prompted Query Programs

These programs ask the user to enter selection criteria, execute an SQL statement to select parcels based on the criteria, and display the results of the query graphically or textually in the mapping system.
These programs should:

1. If two-way integration functions are available, run the **Setup AUTHORITY Link Program** if it is not executed as part of the mapping system start up.
   If two-way integration functions are not available, establish a connection to the **AUTHORITY** database.
2. Prompt the user to enter parcel selection criteria.
   For example, ask the user to enter the number of registered dogs per property.
3. Execute an SQL statement to the **AUTHORITY** database to select all parcels matching the entered criteria.
4. Optionally, match parcel numbers (**pcl_num**) selected from the **AUTHORITY** database to mapping system parcel identifiers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.
5. If parcels are selected:
   - Plot/shade parcels, and/or,
   - Label the map with **AUTHORITY** data, and/or,
   - Print **AUTHORITY** data in a designated area in the mapping system,
6. If no parcels are selected display a message to say that no parcels matching the criteria were found.

8.1.2 Fixed Query Programs

These programs ask the user to select parcels of interest on the map, execute an SQL statement to retrieve textual data about the selected parcels, and display the textual data in the mapping system.
These programs should:

1. If two-way integration functions are available, run the **Setup AUTHORITY Link Program** if it is not executed as part of the mapping system start up.
   If two-way integration functions are not available, establish a connection to the **AUTHORITY** database.
2. Prompt the user to select parcels of interest.
   Alternately, the last parcel selection may be used.
3. Optionally, match mapping system parcel identifiers selected to **AUTHORITY** parcel numbers.
   This is required if the **AUTHORITY** parcel number is not recorded in the mapping data.

   See *Textual to Spatial Object Integration*.
4. Execute an SQL statement to the **AUTHORITY** database to select desired data for the selected parcel number.
   For example, execute an SQL to select the owner's name, the property address and land title for the selected parcel number.
5. If parcels are selected:
   - Label the map with **AUTHORITY** data, and/or,
   - Print **AUTHORITY** data in a designated area in the mapping system.
6. If no parcels are selected display a message to say that no parcels matching the criteria were found.

8.1.3 Zone Shading Program

Some mapping systems include options to shade each polygon a map using a set of rules. If the mapping system allows rules to be cased on external database attribute values then, a zone map can produced based upon zones recorded in **AUTHORITY**.
If the mapping system does not include a specific function for external attribute rules based shading, then a scripted process to achieve the same result could be written.

8.2 Query Writing

8.2.1 Database Structure

Civica can provide information about where data is stored in the **AUTHORITY** database and how tables are linked on request.

8.2.2 Database Privileges and Synonyms

Privileges and synonyms are usually configured as part of the **AUTHORITY** database installation for tables and views typically used in database query functions and further configuration is generally not required. However, you should contact your Database Administrator to have these privileges checked and modified if required.
If the mapping system uses tables or views in addition to these your Database Administrator must grant SELECT privilege to these for all users running the queries, and for Oracle databases, create a public synonym for each.

8.2.3 Custom View Names

The **AUTHORITY Link** developer may determine that it is preferable to create views in the **AUTHORITY** database for query by mapping functions. If so, views should be name so they do not conflict with the naming conventions used by Civica for **AUTHORITY** database objects. Otherwise, they may be removed by **AUTHORITY** database upgrade procedures.
Any objects created by the mapping system in the **AUTHORITY** database should not be named to begin with:

Object Type	*Authority Naming Convention*
Tables, Views, etc.	au*
	av*
	rt*
	et_*
	ev_*
Indexes	cg*
	stx*

8.2.4

8.2.5 Sample SQL Statements

Following are sample SQL statements for database query functions.
The fields selected, and the **where** clause, should be modified as required. For example, for thematic mapping only the parcel number needs to be retrieved from the **select** statement for input to the mapping system display command, whilst for map labelling the parcel number needs to be included as a condition in the **where** clause of the **select** statement.
For correct parcel resolution the following line of each SQL should be replaced according to the resolution method being used:

```
auprparc.pcl_num = [parcel number]
```

This will ensured parcel numbers (**pcl_num**) in the **AUTHORITY** database are matched correctly with mapping system parcel identifiers.

For **Common Key** resolution, replace:

```
auprparc.pcl_num = [parcel number]
```

with:

```
auprparc.pcl_num in
      ( select pcl_num from aurtvalm
      where val_num in
         ( select val_num from aurtvalm
         where pcl_num = [parcel number] )
```

For **Cross-reference Table** resolution, replace:

```
auprparc.pcl_num = [parcel number]
```

with:

```
auprparc.pcl_num in
      ( select [au_pcl_num_field_name]
      from [xref_table_name]
      where [gis_pcl_num_field_name] =
      [parcel number] )
```

For further information see *Textual to Spatial Object Integration*.

8.2.5.1 Property Data

8.2.5.1.1 Street Address

```
select
  auprparc.pcl_num,
  auprstad.pcl_unt,
  auprstad.unt_alp,
  auprstad.hou_num,
  auprstad.hou_alp,
  auprstad.hou_end,
  auprstad.end_alp,
  auprstad.str_nme,
  auprstad.str_typ,
  auprstad.sbr_nme
from
  auprstad,
  auprparc
where
  auprparc.pcl_num = [parcel number]
  and auprparc.pcl_num = auprstad.pcl_num
  and auprstad.seq_num = 0
-- Only get first address if more than one
```

8.2.5.1.2 Land Title (Formatted)

Hard code or prompt the user to enter Parcel Flag, or do not use one at all.

```
select
  auprparc.pcl_num,
  auprparc.ttl_cde title_type,
  auprparp.ttl_dsc,
  auprparc.fmt_ttl title
from
  auprparp,
  auprparc
where
  auprparc.pcl_num = [parcel number]
  and auprparc.pcl_flg = 'R'
  and auprparc.ttl_cde = auprparp.ttl_cde
```

8.2.5.1.3 Owner Name and Postal Address

```
select
  auprparc.pcl_num,
  rtpostal.fmt_nm1 owner_line_1,
  rtpostal.fmt_nm2 owner_line_2,
  rtpostal.fmt_ad1 address_line_1,
  rtpostal.fmt_ad2 address_line_2,
  rtpostal.fmt_ad3 address_line_3
from
  rtpostal,
  auprparc
where
  auprparc.pcl_num = [parcel number]
  and auprparc.ass_num = rtpostal.ass_num
```

8.2.5.1.4 Property Attributes

This query forms can be used to query any property attribute type (e.g. Land Use, Zoning, etc). The where clause **aupratyp.att_dsc** = can be used to nominate which attribute type should be returned. Either, several queries could be used each hard coding a specific attribute type, or the user could be prompted to select the required attribute type and their selection inserted in the query.

```
select
  distinct auprparc.pcl_num,
  aupratyp.att_dsc att_type,
  aupratyp.cd1_lbl att_code1_label,
  auprattr.att_cd1 att_code1_value,
  auprcode1.att_dsc att_code1_desc,
  aupratyp.cd2_lbl att_code2_label,
  auprattr.att_cd2 att_code2_value,
  auprcode2.att_dsc att_code2_desc,
  aupratyp.cd3_lbl att_code3_label,
  auprattr.att_cd3 att_code3_value,
  auprcode3.att_dsc att_code3_desc,
  aupratyp.cd4_lbl att_code4_label,
  auprattr.att_cd4 att_code4_value,
  auprcode4.att_dsc att_code4_desc
from
  aupratyp,
  [INFORMIX] outer(auprcode auprcode1),
  [INFORMIX] outer(auprcode auprcode2),
  [INFORMIX] outer(auprcode auprcode3),
  [INFORMIX] outer(auprcode auprcode4),
  [ORACLE] auprcode auprcode1,
  [ORACLE] auprcode auprcode2,
  [ORACLE] auprcode auprcode3,
  [ORACLE] auprcode auprcode4,
  auprparc,
  auprattr
where
  auprparc.pcl_num = [parcel number]
  and aupratyp.att_dsc = '[Attribute description]'
  and auprparc.pcl_num = auprattr.pcl_num
```

```
and auprattr.att_typ = aupratyp.att_typ
-- Zone Attribute Code 1
[INFORMIX] and auprattr.att_typ = auprcode1.att_typ
[INFORMIX] and auprattr.att_cd1 = auprcode1.att_cde
[INFORMIX] and auprcode1.att_num = 1
[INFORMIX] and auprcode1.seq_num = 0
[ORACLE] and auprattr.att_typ = auprcode1.att_typ
[ORACLE] and auprattr.att_cd1 = auprcode1.att_cde
[ORACLE] and auprcode1.att_num = 1
[ORACLE] and auprcode1.seq_num = 0
-- Zone Attribute Code 2
[INFORMIX] and auprattr.att_typ = auprcode2.att_typ
[INFORMIX] and auprattr.att_cd2 = auprcode2.att_cde
[INFORMIX] and auprcode2.att_num = 2
[INFORMIX] and auprcode2.seq_num = 0
[ORACLE] and auprattr.att_typ = auprcode2.att_typ(+)
[ORACLE] and auprattr.att_cd2 = auprcode2.att_cde(+)
[ORACLE] and auprcode2.att_num(+) = 2
[ORACLE] and auprcode2.seq_num(+) = 0
-- Zone Attribute Code 3
[INFORMIX] and auprattr.att_typ = auprcode3.att_typ
[INFORMIX] and auprattr.att_cd3 = auprcode3.att_cde
[INFORMIX] and auprcode3.att_num = 3
[INFORMIX] and auprcode3.seq_num = 0
[ORACLE] and auprattr.att_typ = auprcode3.att_typ(+)
[ORACLE] and auprattr.att_cd3 = auprcode3.att_cde(+)
[ORACLE] and auprcode3.att_num(+) = 3
[ORACLE] and auprcode3.seq_num(+) = 0
-- Zone Attribute Code 4
[INFORMIX] and auprattr.att_typ = auprcode4.att_typ
[INFORMIX] and auprattr.att_cd4 = auprcode4.att_cde
[INFORMIX] and auprcode4.att_num = 4
[INFORMIX] and auprcode4.seq_num = 0
[ORACLE] and auprattr.att_typ = auprcode4.att_typ(+)
[ORACLE] and auprattr.att_cd4 = auprcode4.att_cde(+)
[ORACLE] and auprcode4.att_num(+) = 4
[ORACLE] and auprcode4.seq_num(+) = 0
```

8.2.5.1.5 Zoning

If Council has defined the **Attribute Type** and **Code** used for Zoning in the **Property Parameters** program (**i_al069**) then this information can be used to retrieve zoning data instead of using the general Property Attribute query.

```
select
  distinct auprparc.pcl_num,
  auprattyp.att_typ zoning_att_type_code,
  auprattyp.att_dsc zoning_att_type_desc,
  decode(aualparm.sml_no5,1,1,2,2,3,3,4,4)
zoning_att_code_num,
  decode(aualparm.sml_no5,1,auprattyp.cd1_lbl,2,auprattyp.cd2_lbl,3,auprattyp.cd3_lbl,4,auprattyp.cd4_lbl)
zoning_att_code_desc,
  decode(aualparm.sml_no5,1,auprattr.att_cd1,2,auprattr.att_cd2,3,auprattr.att_cd3,4,auprattr.att_cd4)
zone_code,
  auprcode.att_dsc zone_desc
from
  aualparm,
  auprattyp,
  auprcode,
  auprparc,
  auprattr
where
  auprparc.pcl_num = [parcel number]
  and auprparc.pcl_num = auprattr.pcl_num
  -- Attribute Type
  and auprattr.att_typ = aualparm.sml_no3
  -- i.e. Attribute Number used for Zoning
  and auprattr.att_typ = auprattyp.att_typ
  and aualparm.mdu_ref = 'PR' -- Property Parameters
  -- Zone
  and auprattr.att_typ = auprcode.att_typ
  and auprcode.att_num = aualparm.sml_no5
  and
  decode(aualparm.sml_no5,1,auprattr.att_cd1,2,auprattr.att_cd2,3,auprattr.att_cd3,4,auprattr.att_cd4) =
auprcode.att_cde
  and auprcode.seq_num = 0
```

8.2.5.2 Animals

Hard code or prompt the user to enter Tag Set Year, or do not use one at all.

```
select
      auprparc.pcl_num,
      auanmast.ani_num,
      auanmast.ani_name animal_name,
      breed.cde_des breed,
      color.cde_des colour,
      auanmast.ani_dob date_of_birth,
      auanmast.ani_sex gender,
      auantags.ani_tag tag,
      auantags.iss_dte tag_issue_date,
      auanfees.reg_fee registration_fee
from
      auantype,
      auanfees,
      aublcdes breed,
      aublcdes color,
      auanmast,
      [INFORMIX] outer auantags,
      [ORACLE] auantags,
      auprparc
where
      auprparc.pcl_num = [parcel number]
      and auprparc.pcl_num = auanmast.pcl_num
      and auanmast.ani_typ = auantype.ani_typ
      [INFORMIX] and
      auanmast.ani_num = auantags.ani_num
      [INFORMIX] and auantags.tag_set = '[year]'
      [ORACLE] and
      auanmast.ani_num = auantags.ani_num(+)
      [ORACLE] and auantags.tag_set(+) = '[year]'
      and auanmast.brd_no1 = breed.cde_val
      and breed.cde_typ = auantype.brd_val
      and auanmast.col_no1 = color.cde_val
      and color.cde_typ = auantype.col_val
and auanmast.fee_cde = auanfees.fee_cde
```

8.2.5.3 Applications/Registers/Documents

This query can be used to query any document type (e.g. Development Applications, Cemetary Register, Building Certificate, etc). The where clause **audmmprm.doc_dsc =** can be used to nominate which document type should be returned. Either, several queries could be used each hard coding a specific document type, or the user could be prompted to select the required document type and their selection inserted in the query.

```
select
  auprparc.pcl_num,
  audmmprm.doc_dsc document_type,
  audmmast.doc_num document_number,
  audmmast.doc_pre description,
  audmmast.det_cde determination,
  audmmast.det_dte determination_date,
  audmmast.act_off action_officer
from
  audmmprm,
  audmmast,
  auprlink,
  auprparc
where
  auprparc.pcl_num = [parcel number]
  and auprparc.pcl_num = auprlink.pcl_num
  and auprlink.mdu_ref = 'DD'
  and audmmprm.doc_dsc = '[Document Type Description]'
  and auprlink.mdu_fmt = audmmast.fmt_acc
  and audmmast.doc_typ = audmmprm.doc_typ
```

8.2.5.4 Records Management

This query can be used to query any file type (e.g. General files, HR files, etc). The where clause **audmftpr.typ_dsc =** can be used to nominate which file type should be returned. Either, several queries could be used each hard coding a specific file type, or the user could be prompted to select the required file type and their selection inserted in the query.

```
select
  auprparc.pcl_num,
  audmftpr.typ_dsc file_type_description,
  audmfile.key_wrd keyword,
  audmfile.tor_no1 descriptor_1,
  audmfile.tor_no2 descriptor_2,
  audmfile.tor_no3 descriptor_3,
  audmfile.tor_no4 descriptor_4,
  audmfile.tor_no5 descriptor_5,
  audmfile.tor_no6 descriptor_6,
  audmpart.fil_loc file_location,
  audmpart.mov_dte movement_date
from
  audmftpr,
  audmpart,
  audmfile,
  auprlink,
  auprparc
where
  auprparc.pcl_num = [parcel number]
  and auprparc.pcl_num = auprlink.pcl_num
  and auprlink.mdu_ref = 'DM'
  and audmpart.fil_typ = audmftpr.fil_typ
  and audmftpr.typ_dsc = '[File Type Description]'
  and auprlink.mdu_fmt = audmfile.fmt_acc
  and audmfile.fil_typ = audmpart.fil_typ
  and audmfile.fil_num = audmpart.fil_num
  and audmfile.las_prt = audmpart.fil_prt
```

Appendix 1. PC Identifier

The PC Identifier for use in the table **AUALMAPL** should be determined as follows:

1. Get the TCP/IP address of the PC
2. If more than one IP address is returned use the second IP address

   

   Alternately, rather than use the second IP address, include a configuration parameter for the Authority Link allowing the user to specify the unique subnet of the network that the Authority server uses. After returning all IP addresses for the PC simply select the address matching the specified subnet.

3. Determine the constituent octets of the TCP/IP address

   ```
   IP address = Octet4.Octet3.Octet2.Octet1
   ```

   

   Octet positions read right to left

4. Optionally, get the terminal ID (i.e. session ID)
   If you have customers that run the mapping system (and **AUTHORITY**) in a terminal server environment (e.g. Citrix Terminal Server, MS Windows Terminal Server), rather than a local PC environment, you need to complete this step.
   If you do not have customers using a terminal server environment you do not need to complete this step.
   A terminal server session ID can be found using the API for various terminal servers. If you know that only one type of terminal server environment is being used then it is only necessary to use the API for that product. If you know that your customer base is using several different terminal servers then you will need to use several APIs.

5. Determine the PC identifier
   If terminal ID is NOT NULL, let the PC identifier equal:

```
((Octet3 % 100) || (Octet2 % 100) || (Octet1 % 100) ||
terminal_id ) * 10
```

   where % modulus
   || concatenate
   If terminal ID is NULL let the PC identifier equal the TCP/IP address.

Following are 2 examples of setting the PC identifier:

Step	**Citrix Environment**	**Local PC Environment**
1. TCP/IP Address	192.168.106.191 207.50.123.1	192.168.106.191 207.50.123.1
2. TCP/IP Address matching Authority Server network	192.168.106.191	192.168.106.191
3. TCP/IP Octets	Octet 1 = 191 Octet 2 = 106 Octet 3 = 168 Octet 4 = 192	Octet 1 = 191 Octet 2 = 106 Octet 3 = 168 Octet 4 = 192
4. Terminal ID	12	Null
5. PC ID	((Octet3 % 100) || (Octet2 % 100) || (Octet1 % 100) || terminal_id) * 10 = ((168 % 100) || (106 % 100) || (191 % 100) || 12) * 10 = ((68) || (6) || (91) || 12) * 10 = (6869112) * 10 = 68691120	TCP/IP address = 192.168.106.191

Appendix 2. Data Exchange Table (AUALMAPL)

8.3 Field Descriptions

It may be useful to create a copy of the data exchange table **AUALMAPL** for developing and testing the **AUTHORITY Link**. The following table shows the structure **AUALAMPL**.

Column Name	Description	Notes
ip_adr	PC Identifier	Number that identifies PC making request
dir_flg	Direction Flag	Code to indicate system that request is being made to A **AUTHORITY** M Mapping System
pcl_typ	Parcel Type	Type of Parcel that request is being made for 1 Base/subject/include 2 Neighbour/affected
fnc_typ	Function Type	Type of process being requested 1 Display parcels 2 As for **fnc_typ = 1** and initiate neighbour Notification process Used for **dir_flg = M** only 3 Create Authority letter request for parcel Used for **dir_flg = A** only 4 Create neighbour/affected parcel link between Authority application and Authority parcel record Used for **dir_flg = A** only
mdu_ref	**AUTHORITY** Module Type	When **dir_flg = M**, Authority module that initiated request When **dir_flg = A**, Authority module to which request is being sent

Column Name	Description	Notes
fmt_acc	**AUTHORITY** record identifier	Authority key for which request is being made When **mdu_ref = DD**, formatted application number When **mdu_ref = PR**, parcel number
pcl_num	**AUTHORITY** parcel number	Authority parcel number for which request is being made
seq_num	Sequence Number	Sequential number for each grouping of: ip_adr dir_flg pcl_typ fnc_typ pcl_num
ext_typ	**AUTHORITY** external document code	Authority external document (i.e. letter) code Used for **fnc_typ = 3** only
Lay_nme	**GIS LAYER NAME**	Contains the other link type description where the link type has a destination module of MP for Mapping. This may be used for mapping features other than parcels or where the Authority Parcel Number is not in the map. This will be used in conjunction with column gis_ref.
Gis_ref	**GIS REFERENCE**	Contains the GIS reference which is held in Authority using the generic other link functionality for link types set up for Mapping.

8.4 Valid Code Combinations

The following table lists valid code combinations when using **AUALMAPL**.

Column Name	Process						
	To Mapping		From Mapping				
	Display Parcels On Map	Neighbour Notifications	Display Parcels In AUTHORITY	Neighbour Notifications			Property Letter Request
				Application Applicant Letter Request	Application Neighbour Parcel Letter Request	Application Neighbour Parcel Linking	
ip_adr	Variable	Variable	Variable	Variable	Variable	Variable	Variable
dir_flg	M	M	A	A	A	A	A
pcl_typ	1,2	1,2	1	1	2	2	1
fnc_typ	1	2	1	3	3	4	3
mdu_ref	Variable	DD	PR	DD	DD	DD	PR
fmt_acc	Variable	Variable	Variable	Variable	Variable	Variable	Variable
pcl_num	Variable	Variable	Variable	Variable	Variable	Variable	Variable
seq_num	Variable	Variable	Variable	Variable	Variable	Variable	Variable
ext_typ	Null	Null	Null	Variable	Variable	Null	Variable
Lay_nme	Variable	Null	Null	Null	Null	Null	Null
Gis_ref	Variable	Null	Null	Null	Null	Null	Null

Appendix 3. Textual to Spatial Object Integration

Integration between **AUTHORITY** textual objects and their corresponding mapping system objects is achieved by the exchange of object identifiers.
This chapter discusses different ways to achieve object resolution within the scope of the integration model. All integration functions developed within the mapping system must take into account the object resolution method to be used.

8.5 General Data Exchange

All **AUTHORITY** mapping integration functions:

- Send **AUTHORITY** parcel numbers to the mapping system
- Expect to receive **AUTHORITY** parcel numbers from the mapping system

Generally, a single parcel in **AUTHORITY** has a matching parcel in the mapping system. However, in some cases many parcels in **AUTHORITY** relate to only a single parcel in the mapping system. For example, many strata parcels in **AUTHORITY** relate to a single land parcel in the mapping system. **AUTHORITY** attempts to resolve these difference as follows.

Sending parcels to the mapping system

Parcel Type	Parcel number(s) sent to mapping
Strata	Lot 0 parcel of strata plan Ie. Lot 0 parcel with same valuation number as sent parcel
Unreal	Registered parcels on which sent parcel sites Ie. Registered parcels with same valuation number as sent parcel
Proposed	Registered parcels on which sent parcel sites Ie. Registered parcels with same valuation number as sent parcel
Other	Actual parcel number

Receiving parcels from the mapping system

Parcel Type	Parcels selected in AUTHORITY
Strata	All parcels in strata plan (including strata upon strata) Ie. All parcels with same valuation number as received parcel
Unreal	Registered parcels on which received parcel sites and all Unreal parcels on the selected registered parcels Ie. All parcels with same valuation number as received parcel
Proposed	Registered parcels on which received parcel sites and all Proposed parcels on the selected registered parcels Ie. All parcels with same valuation number as received parcel
Other	Actual parcel

When sending/receiving parcel numbers **AUTHORITY** uses the actual parcel number if:

- There is no valuation number link for the parcel number
- The valuation number of the parcel is zero
- The valuation number of the parcel is used by more than one parcel but all parcels with the valuation number are registered
- The valuation number is not current

8.6

8.7 Mapping System Object Resolution

Before processing requests made by **AUTHORITY**, the mapping system must equate the **AUTHORITY** parcel number received from **AUTHORITY** to the mapping system parcel identifier.
Similarly, when sending requests to **AUTHORITY**, the mapping system must equate the mapping system parcel identifier to the **AUTHORITY** parcel number to be sent to **AUTHORITY**.
The **AUTHORITY Link** developer may implement the **AUTHORITY Link** software so as to:

- Hard code the resolution method to be used, or
- Provide a choice of resolution methods, or
- Customise the resolution method for each council upon implementation

8.7.1 Resolution Methods

8.7.1.1 Direct Object Resolution (Common Key)

For direct object resolution the **AUTHORITY** parcel number is recorded in the map data. That is, a common piece of information is recorded in both systems, thus enabling direct reconciliation.

8.7.1.2 Indirect Object Resolution (Cross Reference Table)

For indirect object resolution the **AUTHORITY** parcel number is not recorded in the map data and must be matched to the mapping system parcel identifier each time one system receives a parcel from or sends a parcel to the other system.
This may be achieved using a cross-reference table in either the **AUTHORITY** database or the mapping database.
A cross-reference table would contain the fields similar to the following:

Value	Required
AUTHORITY module type (e.g. PR, AS)	Optional
Unique identifier of the object in **AUTHORITY** (i.e. the parcel number)	Mandatory
Mapping system map or layer	Optional
Unique identifier of the object in the mapping system	Mandatory

The mapping system would use the table to:

1. Determine the set of map objects which a list of parcel numbers received from **AUTHORITY** refers to
   For example,

```
select
      gis_lnk
from
      aurtvalm
where
      pcl_num = [Authority Parcel Number]
```

2. Generate a list of parcel numbers for a set of selected map objects to send to **AUTHORITY**
   For example,

```
select
      pcl_num
from
      aurtvalm
where
      gis_lnk = '[Mapping Parcel Number]'
```

If a cross-reference table is used for object resolution it must be created and managed by the mapping system.

8.7.1.2.1 Cross-reference Tables

A cross-reference is not intentionally provided as part of the **AUTHORITY** database, but several **AUTHORITY** tables can be used as cross-reference tables in addition to their intended use within **AUTHORITY**.

8.7.1.2.1.1 AUALLNKS

The **AUTHORITY** database table **AUALLNKS** is used to record links between records in **AUTHORITY** modules. This table is maintained using **AUTHORITY** update programs for different modules.
A council could define a new **AUTHORITY** module to represent the mapping system (e.g. **MP**) and a new module link type (e.g. **Authority to Mapping**), and then use the **AUTHORITY** program, **Parcel Maintenance** (**i_pr038**), to record relationships between **AUTHORITY** parcel numbers and mapping parcel identifiers. Hence, this table could be used as a cross-reference table.

See *Note* at the end of this chapter.

8.7.1.2.1.2 AURTVALM

The **AUTHORITY** database table **AURTVALM** is used to record links between parcels, rate assessments, valuation numbers and third party records. This table is maintained using the **AUTHORITY** program, **Valuation Link Maintenance** (**i_rt066**).
A council could enter mapping parcel identifiers as the third party identifier in this table, and thus, record relationships between **AUTHORITY** parcel numbers and mapping parcel identifiers. Hence, this table could be used as a cross-reference table.

8.7.1.2.1.3 AUPRPARC

The **AUTHORITY** database table **AUPRPARC** is used to record parcel information. This table is maintained using the **AUTHORITY** program, **Parcel Maintenance** (i_**pr038**).
The table contains several user-defined fields for entry of custom council data. A council could enter mapping parcel identifiers in a designated user-defined field in this table, and thus, record relationships between **AUTHORITY** parcel numbers and mapping parcel identifiers. Hence, this table could be used as a cross-reference table.

8.7.2 Mapping System Parcel Number

The following applies only if the **Common Key Resolution** method is used for parcel resolution.

Common Key Resolution means that the **AUTHORITY** parcel number is entered in the mapping data.

In the mapping system, the following **AUTHORITY** parcel number should be recorded on strata, unreal and proposed parcels:

Parcel Type	Parcel Number of
Strata	Strata Plan Lot 0 parcel
Unreal	Registered land parcel on which parcel sits
Proposed	Registered land parcel on which parcel sits

8.7.3 Object Matching

Civica do not provide services to insert the **AUTHORITY** parcel number in the mapping data, or to create a cross-reference table matching **AUTHORITY** parcel numbers and map tags.

The **AUTHORITY Link** developer may provide this service for council at their discretion, or where they do not wish to do so, Civica can suggest third parties that provide this service.

If the **AUTHORITY Link** developer offers this service please ensure Council staff are aware that they are responsible for ongoing parcel matching as parcels are added to and deleted from **AUTHORITY** and the mapping system.

8.8

8.9 Note

In **AUTHORITY** 5.0 and earlier, **AUTHORITY** sends and receives parcel numbers to and from mapping systems, and thus, parcel resolution with mapping system parcel identifiers must be performed by the mapping system.

From a future, yet to be confirmed release, onwards, upon setting of a designated system parameter, **AUTHORITY** mapping integration functions will:

- Send mapping system parcel identifiers to the mapping system
- Expect to receive mapping system parcel identifier from the mapping system and then resolve which **AUTHORITY** parcels these represent

Thus, parcel resolution with mapping system parcel identifiers will be performed by **AUTHORITY**.

Council will therefore have the ability to choose between **AUTHORITY** parcel resolution and mapping system parcel resolution.

AUTHORITY parcel resolution will be achieved through the creation and management of a cross-reference table as part of the **AUTHORITY** database. The existing **AUTHORITY** table, **AUALLNKS**, will be used as the cross-reference table.

The revised model will also expand integration capabilities to enable integration between objects that can be represented spatially apart from parcels, such as assets. For example, **AUTHORITY Property** records may have a relationship defined between the **AUTHORITY** parcel number and the identifier of the corresponding polygon on the cadastral map layer. Whereas **Asset** records may have a relationship defined between the asset number and the identifier of the corresponding point or line or polygon on the asset map layer.

Appendix 4. Running AUTHORITY Programs

8.10 ulaunch

ulaunch is a program installed on the PC as part of the **AUTHORITY** menu processor. It is used to start **AUTHORITY** programs from the PC shell rather than by clicking on items within the **AUTHORITY** menu processor. Thus, it can be used by external systems, such as the mapping system, to start **AUTHORITY** programs.

8.10.1 Syntax

The **ulaunch** command uses the same base syntax for all **AUTHORITY** programs:

```
ulaunch /f [alias] [filters]
```

Where: `alias` **AUTHORITY** alias for program

`filters` Filters for parcels to be displayed

Authority Program LIST lists the default alias and filters for **AUTHORITY** programs used for mapping integration. Civica can provide parameters for additional programs on request.

Alias

Default aliases are defined for many but not all programs. Where an alias is not defined for a program, if the program is to be started using **ulaunch**, then an alias must be defined. Council may also change aliases as desired.
Therefore, as program aliases are variable it is recommended that the alias for any **ulaunch** commands used in the **AUTHORITY Link** are not hard coded. A configuration file could be included so the user can enter each alias to match that defined in **AUTHORITY** or an SQL statement can be used to automatically set aliases to those defined in **AUTHORITY.**

The following SQL statement can be used to determine the alias used by council for a specific program:

```
select
        m.als_ref
from
   aumnmenu m,
        aumncmnd c
```

```
where
        m.mnu_num = c.mnu_num
        and c.cmd_lne like '[command]'
```

Authority Program LIST lists the command for **AUT**HORITY programs used for mapping integration.

8.10.2 Example

This is an example of using **ulaunch** to display parcels selected in the mapping system in the **AUTHORITY Rates Enquiry** program (**e_rt007**):

API command `ulaunch /f [alias] [filters]`

Alias `rtenq`

Filters `-f pcl_num [parcel numbers]`

The mapping system would execute:

```
ulaunch /f rtenq pcl_num 1,2,3,4
```

In this example, the mapping system needs to insert the parcel numbers based on the parcels selected in that system.

8.11 Web Browser Mapping Systems

Web browser based applications are designed to be self-contained, operating only in the browser, rather than to be integrated with other desktop software. As such, it is recommended that the mapping software to be integrated with the non-browser, desktop based **AUTHORITY** software, is also be non-browser, desktop based.

Due to the nature of web browser applications, commands executed by the browser application are processed on the web server, and not the PC running the browser. The **AUTHORITY ulaunch** command, however, must be executed on the PC on which the **AUTHORITY** menu system is running.

To run **ulaunch** from a web browser application, instead of executing the **ulaunch** command directly:

1. Write the **ulaunch** command to a file with the extension BAT. The file should be located so that the web server can access it.
2. Sent the file from the web server to the web browser.

The browser will prompt the user to open the file using the application defined for the file type BAT on the PC. This is comparable to when a web page contains a link to an MS Word file or Adobe Acrobat document, and the PCs file type definitions are used to open the file. A file extension of BAT results in contents of the file being executed on the PC. Therefore, when the **ulaunch** BAT file is sent to the PC, its contents, the **ulaunch** command, will be run on the PC.

Appendix 5. AUTHORITY Program List

Module	Program				
	Description	Number	Command	Default Alias	Filters[1,2]
Process AUALMAPL					
Property	Process Mapping Requests	i_al033	%i_al033.4gs	gisreq	None
Start Property Bulk Update					
Property	Property Bulk Update	i_pr079	%i_pr079.4gs	None	None
Display Parcels					
Animals	Animal Enquiry	e_an005	%e_an005.4gs	anenq	-f pcl_num [parcel numbers] map

1 NOTE! If the AUTHORITY Server is a Windows Server use double quotes (") in filters instead of single quotes (')

2 [parcel numbers] is a comma, separated list of AUTHORITY parcel numbers

Module	Program				
	Description	**Number**	**Command**	**Default Alias**	**Filters**[1,2]
Applications	Applications Enquiry (NZ)	e_dm122	%e_dm122.4gs	None	filter 'auprparc.pcl_num in ([parcel numbers])' map
	Applications Enquiry (SA)	e_dm110	%e_dm110.4gs	None	filter 'auprparc.pcl_num in ([parcel numbers])' map
	Applications Enquiry (Vic.)	e_dm010	%e_dm010.4gs	vicenq	filter 'auprparc.pcl_num in ([parcel numbers])' map
	Certificate Enquiry	e_dm121	%e_dm121.4gs	None	filter 'auprparc.pcl_num in ([parcel numbers])' map
	Construction Certificate Enquiry	e_dm020	%e_dm020.4gs	None	filter "1=1" pcl_num [parcel numbers] map
	Development Enquiry (NSW, Qld, WA)	e_dm120	%e_dm120.4gs	None	filter 'auprparc.pcl_num in ([parcel numbers])' map
Assets	Asset Enquiry	e_as001	%e_as001.4gs	anenq	filter 'auprparc.pcl_num in ([parcel numbers])' map filter 'auasmast.ast_num in ([asset numbers])'
Customer Action Requests	Enquiry	i_dm081	%i_dm081.4gs	caenq	filter 'auprparc.pcl_num in ([parcel numbers])' map
Property	Customer Service Property Enquiry	e_pr005	%e_pr005.4gs	prenq5	filter 'auprparc.pcl_num in ([parcel numbers])' map
	Parcel Maintenance	i_pr038	%i_pr038.4gs	prmast	filter 'auprparc.pcl_num in ([parcel numbers])'
	Property Enquiry	e_pr001	%e_pr001.4gs	prenq	filter 'auprparc.pcl_num in ([parcel numbers])' map
Rates	Enquiry	e_rt007	%e_rt007.4gs	rtenq	-f pcl_num [parcel numbers] map
	Rates Bulk Update	i_rt145	%i_rt145.4gs	rtbulk	-f pcl_num [parcel numbers]
Records Management	Correspondence Enquiry	e_dm003	%e_dm003.4gs	dcenq	filter '1=1' pcl_num [parcel numbers] map
Registers	Generic Document Enquiry	e_dm016	%e_dm016.4gs	ddenq2	filter 'auprparc.pcl_num in ([parcel numbers])' map

Appendix 6. Table Structures

8.12 AUALLNKS – Module to Module Links

Only those fields pertinent to mapping integration are listed.

Column Name	Type	Null	Description
src_mdu	char(2)	no	**AUTHORITY** module type (e.g. PR, AS)
src_acc	char(50)	no	Unique identifier of object in **AUTHORITY**
lnk_typ	smallint(5)	no	Link type
des_mdu	char(2)	no	Identifier for system that object is linked to (e.g. MP for mapping system)
des_acc	char(50)	no	Unique identifier of the object in other system

8.13 AUALMAPL – Mapping Data Exchange

Column Name	Type	Null	Description
ip_adr	char(15)	no	PC Identifier
dir_flg	char(1)	no	Direction Flag
pcl_typ	smallint(5)	no	Parcel Type
fnc_typ	smallint(5)	no	Function Type
mdu_ref	char(2)	yes	**AUTHORITY** Module Type
fmt_acc	char(22)	yes	**AUTHORITY** record identifier
pcl_num	integer(10)	yes	**AUTHORITY** parcel number
seq_num	smallint(5)	yes	Sequence Number
ext_typ	char(10)	yes	**AUTHORITY** external document code

8.14 AUALPARM – AUTHORITY Parameters

The purpose of fields in this table varies depending on **AUTHORITY** Module. Only those relevant to the **Property** module are listed. Only those fields pertinent to

mapping integration functions are listed; those fields discussed in this document are highlighted in bold.

Column Name	Type	Null	Description
mdu_ref	char(2)	no	Authority module Only **PR** is used by mapping integration functions
chr_no1	char(10)	yes	**Bulk Update Import Path (Part 1)**
chr_no2	char(10)	yes	**Bulk Update Import Path (Part 2)**
chr_no3	char(10)	yes	**Bulk Update Import Path (Part 3)**
chr_no4	char(10)	yes	**Bulk Update Import Path (Part 4)**
chr_no5	char(10)	yes	Notification Report (Part1)
chr_no6	char(10)	yes	Notification Report (Part 2)
chr_no7	char(10)	yes	Notification Report (Part 3)
chr_no8	char(10)	yes	Notification Report (Part 4)
chr_no9	char(10)	yes	Notification Report (Part 5)
dsc_no1	char(30)	yes	**Command** Used when Map Product = 4, Latitude
dsc_no4	char(30)	yes	Workspace Not implemented
dsc_no5	char(30)	yes	Location Not implemented
dsc_no6	char(30)	yes	Column Name Not implemented
dsc_no7	char(30)	yes	Table Name Not implemented
flg_no3	char(1)	yes	Notify Occupier
flg_no4	char(1)	yes	Generate Multiple Letter
flg_no5	char(1)	yes	Use mapping linkage table Available from Authority 4.7 onwards
sml_no1	smallint(5)	yes	**Map Product** **None No integration in use** **2 Integration using AUALMAPL and method described in Developer's Guide** **4 Integration with Latitude** **Although other apparently valid mapping systems are listed these integration functions will not work when Map Product is set to these.**
sml_no2	smallint(5)	yes	Title Type
sml_no3	smallint(5)	yes	Zoning Type
sml_no4	smallint(5)	yes	Instrument
sml_no5	smallint(5)	yes	Zoning Code
sml_no6	smallint(5)	yes	Parish
sml_no7	smallint(5)	yes	County
igr_no1	integer(10)	yes	Title Memo Type (PR)

Column Name	Type	Null	Description
igr_no2	integer(10)	yes	Title Memo Type (DD)
igr_no3	integer(10)	yes	Address Memo Type (PR)
igr_no4	integer(10)	yes	Address Memo Type (DD)

8.15 AUDMEXTP – External Document Types

Only those fields pertinent to mapping integration are listed.

Column Name	Type	Nullable	Description
ext_typ	char(10)	no	External document type code Entered in AUALMAPL when fnc_typ = 3
ext_dsc	char(70)	no	External document type description Could be listed in selection list presented in mapping system for fnc_typ = 3
mdu_ref	char(2)	no	Authority module that external document is used for
for_doc	smallint	yes	Authority application type that that external document is used for

8.16 AUMNCMND – Menu Item Commands

Only those fields pertinent to mapping integration are listed.

Column Name	Type	Null	Description
mnu_num	integer(10)	no	Menu number
cmd_lne	char(70)	no	Command line text

8.17 AUMNMENU – Menu Items

Only those fields pertinent to mapping integration are listed.

Column Name	Type	Null	Description
mnu_num	integer(10)	no	Menu number
als_ref	char(8)	yes	Alias reference

8.18 AUPRPARC – Property Parcels

Only those fields pertinent to mapping integration are listed.

Column Name	Type	Null	Description
pcl_num	serial(10)	no	**AUTHORITY** parcel number
Uda_fl[field number]	char(8)	yes	User definable alpha field (e.g. Unique identifier of the object in the mapping system)

8.19 AURTVALM – Valuation Number Links

Column Name	Type	Null	Description
val_num	char(15)	no	**AUTHORITY** valuation number
ass_num	integer(10)	no	**AUTHORITY** assessment number
pcl_num	integer(10)	no	**AUTHORITY** parcel number
gis_lnk	char(20)	yes	Unique identifier of the object in the mapping system

Adelaide	+ 61 8 8364 6111	**Sydney**	+ 61 2 8324 3000
Melbourne	+ 61 3 8676 4400	**Toowoomba**	+ 61 7 4639 3500
Newcastle	+ 61 2 4941 9400		
Orange	+ 61 2 5310 2300	**Auckland**	+ 64 9 929 4590
Perth	+ 61 8 9367 6111	**Christchurch**	+ 64 3 281 8092

info@civica.com.au
www.civica.com.au