Weather Tunnel

Version	**Final v1.0**
Project	**Architecture**
Create Date	**20 January 2012**
Update Date	**13 January 2017**
Author(s)	**Kevin Rogers**

Overview

What is the Weather Tunnel?

Meteorology is the study of how energy effects the atmosphere. The Weather Tunnel is a simple application, meant to run on a portable device and display measurements of atmospheric effects at specific times and locations. The technology to measure atmospheric effects is currently impractical to carry in a portable device, so the data is gathered from a network of distributed sensors (stations) and made available to the application over the Internet. This approach makes the current conditions of any location dependent on both the quality of the station and distance from the station. In the case of distance, an increased number of weather stations will generally improve accuracy by shortening distances. Also, using the network allows for any location to be tracked and additional, climate related information to be supplied.

The primary local atmospheric data delivered by the Weather Tunnel:

Temperature:	The amount of energy in the air. Most popular measure of current conditions.
Wind Speed:	The speed and direction the air is circulating.
Water Level:	The amount of water held in the air. Affects the feel of the air from dry to muggy.
Perceived Temp:	A combination of temp., wind speed and water level that affects how hot/cold the air feels.
Air Pressure:	The density of the air. Sometimes a signal of coming weather conditions.
Historic Avg.:	The average temperature range for a specific day.
Precipitation:	Recent measure of rain/snow fall.
Conditions:	A general evaluation of the visible conditions, e.g. sunny, cloudy, rainy, etc.

The secondary local data delivered by the Weather Tunnel:

Location:	Where are these conditions in effect. Includes place name and elevation.
Time:	What is the local time. What is the last observation time.
Label:	Any special information about the location.
Season:	One of the four segments of the year based on the earth's position in its solar rotation.
Stations:	How far away from the location are the measurements being taken.
Alerts:	Some countries provide weather based warning Alerts.

The forecast data delivered by the Weather Tunnel:

Temperature:	The range of temperature for the day.
Wind:	The expected average wind speed for the day.
Precipitation:	Percentage chance of rain/snow.
Moon Phase:	The moon phase for that night.
Sun Rise/Set:	The time of sun rise and sun set for that day.
Time:	What is the last forecast time.
Condition:	General evaluation of the expected visible conditions.

Why do this?

The current set of applications for mobile devices that provide weather data is not great. Most of the weather information is from the free, public stations. All the applications focus on advertisements and driving viewers to videos and their other media (TV/Web). They have overly fancy graphics that add nothing to the raw data presented. They either do not provide very much information or just dump what they have on the screen.

A better application will get more relevant data from private weather stations in addition to public ones. The application will focus on providing the weather in a quickly readable form, not on videos/audio or redirects to other media. This application will have all the raw data, but focus showing a more understandable, clean version of it.

Model

Requirements

GUI

The user interface will consist of three (5) primary data areas: Slate, Detail, Forecast, Alerts and Widgets.

The Slate is a list of all locations a person wants to track. The list has a small amount of information about each location for quick review. The weather condition icon / text, location name/alias, time and temperature. If there is an available Alert, a small icon appears. The Slate is where a person can search out new locations, and the settings menu on the Slate is where all application settings are adjusted. The Slate will be scrollable to allow access to a list of up to 25 locations. Information on the Slate will be updated periodically. NOTE: The top location on the Slate is always the current location and cannot be deleted (although location tracking can be turned off).

A single tap of a line on the Slate will bring up the second component, the Detail. The Detail is a scrollable screen that shows all the weather information available for the location. This includes the a description of the current conditions, raw data, current day forecast information and other items like sunrise and sunset times. Bringing up a Detail window will refresh the data for that location on other screens. Detail is one of multiple screens in a tabbed display.

The other tabbed screen is the Forecast. On this screen is displayed 10 days of expected conditions, high / low temperatures and wind speed. If any rain or snow is expected, a probability (10% - 100%) and expected amount is also displayed. The copy describing the specific day's forecast is available by tapping on the individual forecast day. Forecast is one of multiple screens in a tabbed display.

The Alerts screen displays one or more alerts posted about a specific area. The location, title, expiration and detail of the alert is shown. If there are more than one alert, a navigational aid will appear (next/back button) to allow display each alert. The Alert screen is accessible from the tabbed Detail or Forecast screens in the Action Bar. Note that alerts are typically only available for North American and European locations.

The last component are Widgets. They are standalone, 2x1 rectangular components that can be placed anywhere on the desktop. Widgets are made up of a fixed image that reflects the current condition, the location name/alias and the current temperature. If there is an available Alert, a small icon appears.   The current condition image will relay day and night, clear, clouds, rain, lightning and snow. Widgets can be selected from any location already on the Slate, but must be explicitly deleted (removal from the Slate will not remove it from the desktop). A tap on the Widget will bring up the Detail screen and allow navigation back to the Slate. The data on a Widget reflects what is on the Slate.

Services/DB/WS*

Each location added to the Slate creates a location record in the local database. This contains information about the location and the last requested weather information. A service runs in the background that periodically updates these records. This allows for the GUI elements to be responsive to user requests independent of the network response times.

The Web Services are REST based and return XML formatted data. This data will be parsed into an internal data structure and used all over the application. There are a number of Web Services used. The majority of the calls will be to a Weather Tunnel Web Service, which in turn is calling the Weather Underground Web Service.

Data

The weather data will come from the Weather Underground (WU). Their data contains not only data from public weather stations but also from a network of over 20,000 private stations. This leads to more accurate current condition data. The weather station data cost money based on transactions, which could become prohibitive with a large user base. To avoid scale costs base on users, a middle man Web Service will exist to cache weather station data. This will limit requests back into the WU by just performing updates for stale weather stations. For example, instead of a million users making a million requests to the WU, the million customers would make a maximum of 75K requests per update period. The data cache should also speed up data retrieval, too. Non-weather station requests like search will just be passed through, but should be significantly lower in number.

Security

To allow proper monitoring and some bit of control over access, the Weather Tunnel API (WT-API) will require registration and the passing of a ticket when making calls. The registration will consist of generating a GUID, saving that on the system and returning it to the WT-GUI. The WT-GUI will then pass that GUID back as a “ticket” upon each future call. This will allow an auditing system to be added based on the GUID and provide some visibility on individual device’s use of the system.

Localization (dynamic?)

To support International use, the application will be able to support multiple languages. The use of an online translation service to do this dynamically is preferred, given the limited text.

Design

The use of meteorological data from the Weather Underground (WU-API) poses two primary problems: 1) it is expensive and 2) it is big and occasionally non-responsive. To aid in managing both of these problems, a Weather Tunnel Web Service (WT-API) acts as an intermediary. The primary reason a cache strategy works in this situation is that the number of requests is expected to far exceed the (reasonable) number of data points. Basically, at any given time there are only ~75K data points, while there could be significantly more user requests at the same time. Admittedly, the time window has to be somewhat wide (minutes), but most weather condition deltas are quite gradual. Also, given the limited number of sensors, most data is going to be an estimate of the actual location. That is, location distance is probably a greater cause of error than time over a few minutes.

The WT-API supports a pass through for searching, which is a reasonably sparse activity. For station data queries, the Web Service provides a caching mechanism that both proxies and governs the direct calls the WU data. The proxy acts to adjust the pricing model to a more affordable per station versus the potentially expensive per user. It also protects the client from most WU-API outages, speeds access and allows molding of the data to optimal shapes. It ultimately will provide additional, non WU data within the stream, too. Secondarily, it acts as a governor to limit the cost of the WU-API data to a predictable amount, keeping use within the agreed upon boundary conditions set up in the terms of use, e.g. max requests per day and max per minute.

Architecture

The system will b set up as a horizontally scalable set of Web Services, with each machine also running a standalone cache. The Web Service interface will be customized to the needs of the Weather Tunnel application, and will use the cache to retrieve any weather station data. If a station is not found, the interface will fall back to directly calling the WU-API. As station data ages, it will be refreshed by a separate queue based thread. Any searches for stations will be passed directly to the WU-API.

For the short term, a 2 machine configuration utilizing a single shared memory cache (combining extra RAM on each box) is expected to be capable of supporting the initial launch of the Weather Tunnel. For the longer term, doubling the capacity and/or number of boxes should support full utilization. Currently the MSFT Azure Cloud Platform has a single cache solution in Beta Testing. In the meantime, the Memcached product will be used..

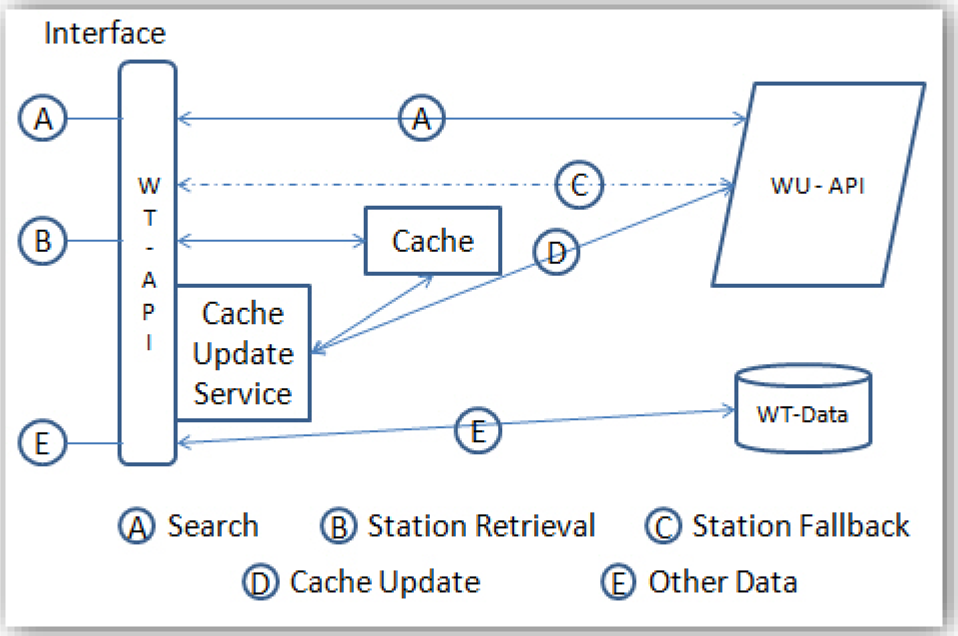

Figure 1

In the Search Interface (A), the request is passed along as is to the WU-API. Search here is primarily a function of city name and would be impractical to cache. Search is not expected to be a large percentage of the overall requests.

In the “B” interface, the Web Service will request the data directly from the Cache. If the data exists it is returned to the user. If the data does not exist, a direct call to the WU-API is made and the data is both returned to the user and added to the Cache. This has the potential to exhaust the available per-minute request allocation from the WU-API in some circumstances (e.g. Cache is lost). Along with Search and the Cache Update Service, all calls to the WU-API will be capped to make sure they stay within contractual limits.

Part of a request from the Cache includes keeping track of the data it is returning to check if it is stale. Staleness is based on some predetermined configuration. If the data is stale, it is added to an update queue. The update queue is serviced by the Cache Update Service (CU). The CU monitors the update queue and updates the CS from data retrieved from the WU-API. The CU takes into consideration the current contractual request limits of the WU-API (per minute & per day) to make sure they are not exceeded.

Security

To protect against abuse of the WT-API, a very soft authentication will be instituted. Prior to the first time the Weather Tunnel tries to use the API, it will register itself. Registration will be the passing of some unique bit of data to the server, and in return a GUID will be supplied. After that the GUID will be required to use the WT-API. Each WT-API method call will write out an Audit record, which includes the GUID. This will allow monitoring of use from specific callers. If a caller appears to be abusing the system, they can be deactivated. While not a specifically tight security mechanism, it is easy for the user, as there is nothing they have to do.

Implementation

Interfaces

Climate Inquiry Web Service

Namespace	http://coolftc.org/CoolftcWTClimate
Proxy	Coolftc.WTClimate
Class	Climate
URL	http://*LM*/CoolftcMET/Climate.svc/rest :: for use w/ REST (xml). Note: When using REST, all data is returned as xml. http://*LM*/ CoolftcMET/Climate.svc/basic :: for WS-I v1.1 compliant. http://*LM*/ CoolftcMET/Climate.svc/ws :: for WS-I v2.0 compliant. http://*LM*/ CoolftcMET/Climate.svc?wsdl :: for WSDL file. *LM* is the host name.
Description	The Climate API provides an interface to retrieve climate and location information. The Climate API creates an aggregation point for multiple data sources. It provides faster and more cost effective data than going directly to the sources from the client device.

Methods	**GetRegistration**	:Set up user with a unique id for use as a ticket.
	GetWeather	:Return meteorological information for the requested location.
	GetWeatherDL	:Return non-forecast meteorological information for the requested location.
	GetWeatherSL	:Return summary of meteorological information for the requested location.
	GetLocation	:Return location information for the requested locality.
	GetPrepaqList	:Return all the destinations + coordinates for the prepaq.
	Version	:Returns the Version of the Web Service Class.

GetRegistration

Description	Set up user with a unique id for use as a ticket.	
Signature	**GetRegistration**(`string ticket, string unique`) **REST**: v1/weather/register/{ali}?ticket={ticket}	
Return Type	`XmlElement`	
Parameters	`ticket`	The security token used to authenticate the application.
	`unique`	Some unique item of data from the device.
Dependencies		
Return	Returns a GUID to be used as the ticket for all other WS calls.	
Remarks		

GetWeather

Description	Return meteorological information for the requested location.
Signature	**GetWeather**(`string ticket, string station`)

	REST: v1/weather/{station}?ticket={ticket}
Return Type	`XmlElement`
Parameters	`ticket` The security token used to authenticate the application. `station` The weather station to from which to retrieve data.
Dependencies	
Return	Returns an xml document that contains the data for a weather station.
Remarks	If the Cache cannot return a value and the maximum number of requests to the WU-API have been made, an exception will be thrown.

GetWeatherDetail

Description	Return non-forecast meteorological information for the requested location.
Signature	**GetWeatherDetail**`(string ticket, string station)` **REST**: v1/weather/detail/{station}?ticket={ticket}
Return Type	`XmlElement`
Parameters	`ticket` The security token used to authenticate the application. `station` The weather station to from which to retrieve data.
Dependencies	
Return	Returns an xml document that contains a subset of the data for a weather station. The data returned is the same as the GetWeather(), except missing the <forecast> node. The <forecast> node is typically 80%-90% of the returned data.
Remarks	If the Cache cannot return a value and the maximum number of requests to the WU-API have been made, an exception will be thrown.

GetWeatherSL

Description	Return summary of meteorological information for the requested location.
Signature	**GetWeatherSL**`(string ticket, string station)` **REST**: v1/weather/slim/{station}?ticket={ticket}
Return Type	`XmlElement`
Parameters	`ticket` The security token used to authenticate the application. `station` The weather station to from which to retrieve data.
Dependencies	
Return	Returns an xml document that contains a subset of the data for a weather station. The data returned is a single string that can be parsed by using split with a question mark (?) delimiter. Example of returned data: -<response><sl>23?Clear?clear?1?0?0?Europe/London</sl></response> 23 – Temperature Clear – Descriptive Condition clear – Condition Icon 1 – Day (0) or Night (1) 0 – Current data (0) or Stale data (1) 0 – No Alerts (0) or Has Alerts (1) Europe/London – Time zone.
Remarks	If the Cache cannot return a value and the maximum number of requests to the WU-API have been made, an exception will be thrown.

GetLocation

Description	Return location information for the requested locality.	
Signature	**GetLocation**`(string ticket, string link)` **REST**: v1/location/search/{link}?ticket={ticket}	
Return Type	`XmlElement`	
Parameters	`ticket`	The security token used to authenticate the application.
	`link`	The data needed to do a search for a specific location. Usually in the form of "zmw:94960.1.99999" or "longitude,latitude".
Dependencies	This is a pass through for the WU-API geolookup.	
Return	Returns an xml document that contains the data for a location. The critical data returned is the list of public and private weather stations in proximity to this actual location. These stations are the key used in GetWeather().	
Remarks	If the the maximum number of requests to the WU-API have been made, an exception will be thrown.	

GetPrepaqList

Description	Return all the destinations + coordinates for the prepaq.	
Signature	**GetPrepaqList**`(string ticket, string prepaq)` **REST**: v1/location/prepaq/{prepaq}?ticket={ticket}	
Return Type	`XmlElement`	
Parameters	`ticket`	The security token used to authenticate the application.
	`prepaq`	The key used to look up a collection of related destinations.
Dependencies	The Prepaq data is a locally owned database of Destinations and Coordinates.	
Return	Returns an xml document that contains a list of Destinations and their Coordinates. For example, a Prepaq of North American Airports would contain all the International Airport s in the US & Canada, including their longitude and latitude.	
Remarks		

Version

Description	Returns the Version of the Web Service Class.
Signature	**Version()** **REST**: v1/utility/version
Return Type	`string`
Parameters	`n/a`
Dependencies	
Return	Gets the version of the Web Service.
Remarks	Comes in the form: major.minor.revision.build, e.g. 1.3.0.1

Climate Maintenance Web Service

Namespace	http://coolftc.org/CoolftcWTSupport
Proxy	Coolftc.WTSupport
Class	Support
URL	http://*LM*/CoolftcMET/Support.svc/rest :: for use w/ REST (xml).
	http://*LM*/ CoolftcMET/Support.svc/basic :: for WS-I v1.1 compliant.
	http://*LM*/ CoolftcMET/Support.svc/ws :: for WS-I v2.0 compliant.
	http://*LM*/ CoolftcMET/Support.svc?wsdl :: for WSDL file.
	LM is the host name.
Description	The Support API provides an interface to perform maintenance and get metadata for the Application. The methods supported in this interface are not used by the Weather Tunnel application, but can affect the data returned by the Climate Interface.

Methods	**NewDestination**	:Add a destination to a prepaq.
	DelDestination	:Delete a destination from a prepaq.
	GetRegInfo	:Return the information for a specific registration.
	GetRegByDate	:Returns a list of registration ids for the date range.
	ChgRegActive	:Change the activation on a registrations. Usually to deactivate.
	GetLogList	:Return a list of all log items from the ledger.
	GetLogFilter	:Return a list of specific log items from the ledger.
	DelLogItem	:Delete a specific log item.
	DelLogFilter	:Delete log items found within a date range.
	GetStats	:Return use statistics for the Web Tunnel.
	Version	:Returns the Version of the Web Service Class.

NewDestination

Description	Add a destination to a prepaq.	
Signature	**NewDestination**(`string ticket, string prepaq, string name, string where`)	
	REST: v1/location/prepaq/new/{prepaq}/{name}?where={where}&ticket={ticket}	
Return Type	`n/a`	
Parameters	`ticket`	The security token used to authenticate the application.
	`prepaq`	The key used to look up a collection of related destinations.
	`name`	The name of the Destination.
	`where`	The GPS Coordinates of the Destination.
Dependencies	The Prepaq data is a locally owned data base of Destinations and Coordinates.	
Return		
Remarks	Adds a Destination and its GPS Coordinates to a Prepaq. If the Prepaq does not exist, it will be created.	

DelDestination

Description	Delete a destination to a prepaq.
Signature	**DelDestination**(`string ticket, string prepaq, string name`) **REST**: v1/location/prepaq/del/{prepaq}/{name}?ticket={ticket}
Return Type	`n/a`
Parameters	`ticket` The security token used to authenticate the application. `prepaq` The key used to look up a collection of related destinations. `name` The name of the Destination.
Dependencies	The Prepaq data is a locally owned data base of Destinations and Coordinates.
Return	
Remarks	Deletes a Destination and its GPS Coordinates from a Prepaq.

GetRegInfo

Description	Return the information for a specific registrations.
Signature	**GetRegInfo**(`string ticket, string key`) **REST**: v1/utility/register/{key}?ticket={ticket}
Return Type	`string`
Parameters	`ticket` The security token used to authenticate the application. `key` The registration key of interest.
Dependencies	
Return	The information about the key. This is returned as a single, comma delimited string in the form of: key,activation,unique.
Remarks	If more than one registration key exists, additional comma delimited values are appended, delimited by question marks.

GetRegByDate

Description	Returns a list of registration ids for the date range.
Signature	**GetRegByDate**(`string ticket, string start, string end`) **REST**: v1/utility/register/search?start={start}&end={end}&icket={ticket}
Return Type	`RegistrationID []`
Parameters	`ticket` The security token used to authenticate the application. `start` The beginning date used to create a date range. `end` The end date used to complete the date range.
Dependencies	
Return	Returns all the registration records in the database for a specific date range.
Remarks	Dates expected to be in ISO 8601 format: yyyy-MM-ddTHH:mm:ss.fffffffzzz (where zzz = +00:00)

ChgRegActive

Description	Change the activation on a registrations. Usually to deactivate.
Signature	**ChgRegActive**(`string ticket, string key, boolean act`) **REST**: v1/utility/register/chg/{key}?active={act}&icket={ticket}
Return Type	`n/a`
Parameters	`ticket` The security token used to authenticate the application. `key` The registration key of interest. `act` The activation value. True is active, False is deactive.
Dependencies	

Return	
Remarks	While this does update the table, it will only update the in memory data on the machine it runs on. So you may have to try it a few times to get it to stick.

GetLogList

Description	Return a list of all log items from the ledger.	
Signature	**GetLogList**(`string ticket`) **REST**: v1/utility/log?ticket={ticket}	
Return Type	`LedgerLog []`	
Parameters	`ticket`	The security token used to authenticate the application.
Dependencies		
Return	Returns all the log records in the database.	
Remarks		

GetLogFilter

Description	Return a list of specific log items from the ledger.	
Signature	**GetLogFilter**(`string ticket, string start, string end`) **REST**: v1/utility/log?start={start}&end={end}&ticket={ticket}	
Return Type	`LedgerLog []`	
Parameters	`ticket`	The security token used to authenticate the application.
	`start`	The beginning date used to create a date range.
	`end`	The end date used to complete the date range.
Dependencies		
Return	Returns all the log records in the database for a specific date range.	
Remarks	Dates expected to be in ISO 8601 format: yyyy-MM-ddTHH:mm:ss.fffffffzzz (where zzz = +00:00)	

DelLogItem

Description	Delete a specific log item.	
Signature	**DelLogItem**(`string ticket, string key`) **REST**: v1/utility/log/del/{key}?ticket={ticket}	
Return Type	`n/a`	
Parameters	`ticket`	The security token used to authenticate the application.
	`key`	The Row Key of the log entry to be deleted.
Dependencies	Use GetLogList() or GetLogFilter() to obtain a valid key.	
Return		
Remarks		

DelLogFilter

Description	Delete log items found within a date range.	
Signature	**DelLogFilter**(`string ticket, string start, string end`) **REST**: v1/utility/log/del?start={start}&end={end}&ticket={ticket}	
Return Type	`n/a`	
Parameters	`ticket`	The security token used to authenticate the application.

	`start`	The beginning date used to create a date range.
	`end`	The end date used to complete the date range.
Dependencies		
Return		
Remarks	Dates expected to be in ISO 8601 format: yyyy-MM-ddTHH:mm:ss.fffffffzzz (where zzz = +00:00)	

GetStats

Description	Return use statistics for the Web Tunnel.
Signature	**GetStats**(`string ticket`) **REST**: v1/utility/stats?ticket{ticket}
Return Type	`XmlElement`
Parameters	`ticket` The security token used to authenticate the application.
Dependencies	
Return	Returns an xml document that contains information about the use of the Cache.
Remarks	The data returned by this method is only what has been collected during the lifetime of the Cache Service instance. All values are reset upon a recycle of the instance.

Version

Description	Returns the Version of the Web Service Class.
Signature	**Version()**
Return Type	`string`
Parameters	`n/a`
Dependencies	
Return	Gets the version of the .DLL containing the class.
Remarks	Comes in the form: major.minor.revision.build, e.g. 1.3.0.1

Issues

1 **Description.**
How can the phone application code be optimized.

Resolution.
Creating Objects uses memory. Using memory is expensive. Use fewer classes in places needing Optimization. Try to avoid temp object creation (explicit local variables better).
For concatenation use StringBuffer (StringBuilder).
Always use multiple arrays instead of multidimensional arrays.
Use Static Methods to avoid the name redirect of Virtual Methods.
Within a class, always use the private fields directly, as the Virtual Get/Set have a lot of overhead.
When processing in a loop, the Copy/Use of data to local variables is always faster than accessing object fields directly.
Always use a local copy of data in the second clause of a FOR statement, e.g. int x = this.getCnt; for(int i=0;i<x;++i){}
Declare Constants as “final”.
Methods that are final, private or static can be made inline (manually or by the compiler).
ENUMs are really classes and considerably more expensive than Constants.
Most ARMs do not support floating point in hardware (Float/Double), so integers for any heavy lifting with arithmetic.
The SimpleDateFormat class can trigger time zone name resources to be loaded, which takes 1 – 2 seconds. If this is in some sort of loop, significant slowdowns can occur. Later version of Android (ICS) might address this, but always keep an eye out for message “Load time zone name for…” with a Resources tag in the LogCat. Pull it out of the loop and the system does a good job of caching it.
Any work done on bitmaps needs to be power of 2, e.g. resize.

2 **Description.**
How much space (memory) will the Cache Supply Service require?

Resolution.
Per http://www.wunderground.com/about/data.asp, there are a possibly ~70,000 weather stations. If each weather station detail takes up 35K bytes, then that will require ~2500 Megabytes of storage. This will require a “medium” instance if we want to be sure the data will fit. Getting 2 mediums will cost $360, the same price as 2 smalls with a 2 Gig cache service. The cache would keep things up and running and support multiple application instances gracefully, so if we really want this to scale and be available we should go with the AppFabric cache.

3 **Description.**
How much network bandwidth (Mbps) will the Automated calls to the Web Service require?

Resolution.
This depends on the number of users and the number of stored locations per user. Assuming the “Slim” data call will be used primarily (as in automated), each location will require 1 station (50 bytes) every 30 minutes. If each user has 10 locations, that is 0.99K bytes (2*50*10/1024) an hour per user. Or about 0.000275K (.99/60/60) per second per user. Using the small instance, which supports 100 Mbps (or 10,000K bytes / sec), there is enough bandwidth for 36MM users (10000/0.00275). The total amount of data per month is 720K bytes per person. If there are 1MM uses, they will request 687 GB of data (1MM*720/1024/1024). That comes out to $82/month using the $0.12/GB cost of Azure bandwidth.
*Note: This does not estimate user generated (2*35K) requests.
http://blogs.msdn.com/b/avkashchauhan/archive/2011/01/30/windows-azure-network-io-capability-for-each-vm-size.aspx

4 **Description.**
What parts of an application must be measured on the Cloud?

Resolution.
CPU usage. How many transaction can it handle.
RAM usage. Both default system memory and extra specialized memory (cache).
Disk Space. Both total disk space used and number of NURD transactions. This refers to NoSQL storage.
Database. There can be extra cost for the use of a relational database.
Network. Includes single box throughput limits and costs for data moving over the network (both directions).

5 **Description.**
How should invalid data be handled?

Resolution. In General: String Data initialized to zero length string. Numeric data initialized to zero. String Data that needs to communicate to other functions it is not valid/available will be set to “-9999”. String Data that needs to be used directly in the UI that it is not valid/available will be set to “Not Available” or “N/A”. Numeric Data that is found to be not valid/available will be set to -9999. This includes conversion exceptions. In cases where not valid/available Numeric data will just end up being set to zero (0), e.g. wind, POP, etc., it is ok to just set it to zero and skip forcing the code to check for invalid data.	
6	**Description.** How do I add a Favicon to my web page?

Resolution.
Add this to each web page:
`<head profile="http://www.w3.org/2005/10/profile">`
`<link rel="icon" type="image/png" href="http://example.com/myicon.png">`	

7	**Description.** How many calls to the WU-API will be required.
Resolution. Total Stations (ST) * Refresh/Day (RR). ST=20K, RR=32(every 45 min), Calls per Day = 20*32 = 640,000 Max is 70 * 32 = 2,240,000.	
8	**Description.** What are best practices to connection to external web services and databases.
Resolution. Good overview of latest: http://blogs.msdn.com/b/tmarq/archive/2007/07/21/asp-net-thread-usage-on-iis-7-0-and-6-0.aspx	
9	**Description.**
Resolution.	

Backlog

Priority

Create an International Airport Prepaq. ref. http://www.landings.com/evird.acgi$pass*193800883!_h-www.landings.com/_landings/pages/search/search_ap-ident.html better: http://www.airnav.com/airports/ or for world http://worldaerodata.com/ List: http://en.wikipedia.org/wiki/List_of_international_airports_by_country To get specific airport: http://worldaerodata.com/wad.cgi?airport=ksfo
See this for help with parsing HTML (http://worldaerodata.com/wad.cgi?id=US10802&sch=ksfo): http://heckyesmarkdown.com/?u=http%3A%2F%2Fworldaerodata.com%2Fwad.cgi%3Fid%3DUS10802%26sch%3Dksfo&read=1

Create an ability to share (send) weather information. See http://android-developers.blogspot.com/2012/02/share-with-intents.html for Share Intent start. The sent data can include a note plus some details (Location/Temp/Condition/???) plus a link to seeing all the data. There should be an ad suggesting the target get the WT. The button will be on a rounded background that jutes out from the right side of the Detail, above the time. And in the options menu.

Sometimes weather stations can change their name. The API does not know this happens, but forum said not often. More likely a pws has just stopped working very well. Maybe the client should be alerted to "not found" stations and then it can fall back or re-search. Also, recheck that reasonableness checks are in place (no -5000 degrees).

Bugs + Minor Fixes

1. There is no default on date format string in settings, so if not chosen there is odd behavior.
2. The http://www.allareacodes.com/415 has all the NANP data needed to pin point an origination city (although if the person moved and ported the number, all bets are off). Can also cross check with zipworld file. For international, look at using the rate table. This may be best done with the API and doing the conversion from number to city on the server.
3. Setting to allow caching of data to allow off-network access to stored weather info.
4. Support a permanent notification item that can be toggled in settings. Maybe just an icon and show full row when dragged down. Putting a number in the notification bar is not considered good design by google.
5. On the slim results, there can be bad temperature values (-9999) returned.
6. Remove QNF message from log.

Waiting

Review incorporation of more WU Data: Hourly 1-day forecast, Satellite thumbnail, Dynamic Radar image, Tides and Currents, Tides and Currents Raw

Version 2:

Add an 8 point compass rose to show the direction of the wind.

Consider using compression on data stored in cache. Will avoid the problem where it get to be too big.

Provide a Global message that can optionally be displayed on the Slate. It should have a title, body, expiration, ???

Search: Update to use the newer search with 3.0+ SDKs. Create a search without typing – finger gestures + prepaqs. Call it Gesture Search.

Any screen that can be "tapped" to dismiss should say so, but in very small, light lettering.

Create some new Widget designs: Road to Edo, Star Trek. Redo Widget Graphics to support the latest design considerations per UI Guidelines, e.g. 9 patch, version specific layout, etc.

Extra Detail Screen: Add the Current Satellite, Radar map and other images somewhere. The Radar as done by most apps is not perfect. Either lots of network traffic for constant updates or they just get big images and scale them.

Create tool to allow easy analysis of audit records.

Create a Management Client (android?) that allows review of stats and log for last 24 hours.

Add a sidebar settings slide out.

Create some functionality (library or send exception to WT-API) to monitor local health and find bugs in field.

Allow phone number to be used if no address present on Add Friends. Is there an API for this? http://www.melissadata.com/lookups/phonelocation.asp?number=4157356341 http://developer.yp.com/api-overview/Listings%2520Endpoints SOAP API http://webservicemart.com/phone3t.asmx or use TelephoneUtils (geolocal) to get contact location based on phone number. It is a flat file that takes up lots of memory.

Improve handling of the REST 500 return code. Create a custom REST exception class that can store information about failure, and be passed to LogEX.

Improve performance by reducing use of type double and using XPP factory instead of new class.

Extra popup menu item allows editing of Special Text (except for current location).

Support a set of cities for the States Preset. Have data, need a web service. See http://en.wikipedia.org/wiki/List_of_lists_of_settlements_in_the_United_States / http://geolite.maxmind.com/download/geoip/database/GeoLiteCity_CSV/

Put feedback button ability into app to generate future features / bug reports. Perhaps use share Intent or just Email.

Add a barometric icon to the Slate records showing up/down/neutral.

Allow customization of the Detail screen.

Maybe Someday

Support a custom view for horizontal screen layouts on Detail/Forecast. Just put Tab fragments side by side to create horizontal.

World Wide Weather Alerts – check out the WMO for as a possible source http://www.wmo.int/pages/index_en.html. Other disasters might be good too, like earthquakes, tsunami, famine, etc.

Allow resorting of the Slate, e.g. move items up or down.

Extra Detail Screen: Add the Forecast Discussion somewhere.

Social Networking Capabilities:

- Ability to auto generate email/text/tweet (share) current conditions for a place. Include link to web site and/or marketplace link to download app. See the "Share" intent https://plus.google.com/u/0/108967384991768947849/posts/ExqhqWWqSP9 or http://android-developers.blogspot.com/2012/02/share-with-intents.html for Share Intent start.
- Tweet a forecast line.
- Follow specific weather related sites. Follow in twitter but filter only those of concern to app.

Allow Location Prepaqs to be Active/Inactive.

Support multiple languages.

Slate: Allow the "current location" to be added as a permanent location. Maybe in long tap menu.

Slate: Allow the "current location" to be changed if location tracking id turned off.

Provide aggregate weather statistics once all weather stations are in the cache. Example: Hottest/Coldest place on earth. Average temperature based on various filters like latitude, Hemisphere, country, whole earth. Use the "earth marble" image on that screen.

Does EMF have anything to do with the weather? Some phones have a magnet that can be used as a sensor.

Add tides graphic/data to forecast information, both on detail and in 10day.

Improve the constant update of current location. https://plus.google.com/105582563895720345957/posts/APKLe9vBnUc

Improve the use of European Alert information. There is some extra text/field data that is ignored.

Support multiple languages. Hard coded text will need to become loadable. The WU-API can do some of translating of what it provides, while other data can be translated dynamically by Google APIs. Raw string.xml files can be translated (and inserted into the application) using this tool: http://googledevelopers.blogspot.com/2012/03/localize-your-apps-and-content-more.html

Add some current condition suggestion icons/data to top right of Detail. Umbrella+POP, Coat if below average, ice cube if <= freezing

Alerts from the current location should show up in the Notification Bar.

On map, show the public & private weather locations, as well as the specified location.

Add weather "disasters" as part of Climate page. See http://earthobservatory.nasa.gov/Features/ for ideas and source data.

Locations Prepaqs to Add as an API. This specific "type" title will go into the search content provider, but will bring up a list activity built from API call, e.g. Ballparks – North America, Airports – Europe, Ski Resorts – North America, etc.

- Cities in a state
- Largest Cities by population
- Ski Resorts (see Wikipedia list)
- Football use "Stadium Team Name"
- Soccer Stadiums
- National Parks
- Rename Baseball to "BallPark Team Name or Park Name"?
- Golf Courses
- Cricket Fields in England/India/Australia

Add a Global Address Filter so that when someone clicks on an address they are offered to launch the Weather Tunnel (and add it to the Slate). Allow entry of a special message.

If calls to the WU-API are not explosive, possibly implement a 6-hour alarm to keep the Slate up to do date (if no widgets).

Use the notification bar to display the current location's temperature. Have this be defaulted off.

Search finds stations that are not great. Look at Las Vegas and New York City, they pick weather stations that are really bad. The regular weather underground web site does not uses these sites, but picks others as the default. Not sure of the solution.

Done

Switch search to new API (with direct WU calls)

Add a backup PWS to device store.

Switch to new Conditions and Forecast APIs. Call directly at first.

Allow choice of contact location, e.g. work(2), home(1), other(3), custom(0).

Support contacts in non-US countries (requires geolocation). Use Google v3 GeoCoding API.

Replace use of SAX with XPP.

Add “Kevin Speaks” to all API calls.

New Condition Icons based on waves.

Create a Launch Icon (use Clear Condition).

Wind/Humidity should be in forecasts now, check out API data and put Wind Est. in GUI.

Write to log if any try/catch errors trigger. Include class name.

Consolidate Forecast and Condition XPP into one API call.

Write up the Design of the Cache and define the Interface for both the Web Service and Cache.

Build a Web Service and Cache Supply Service in Azure. Support error logging.

Build an Asynchronous Cache Update service to refresh the data as needed.

Build in Snapshot functionality that will let the Cache data persist during a planned shutdown/startup.

Validate any data returned from the WU and do not save invalid data to the Cache.

Create a Stats method that reports on use of the cache, expose as a web service method.

Test Snapshot, small summary, cache stats. Test writing/reading logs from both WS and CS.

Put the Snapshot and Death Thread into the Configuration Change action.

Delete Log needs to allow Application specification. Test deletes across both WS and CS.

Have the automate Slate & Widget update use the Slim API call.

If WT gets data that is returned stale, it should retry soon since it will be getting a refresh at the Cache.

When Application Paused, change refresh time to 90 minutes. When woken up from pause, do a refresh.

Show Temp without decimal (rounded). Make sure all temperature format routines consistent.

Do a location check when the user hits the Refresh button.

Add WU 1-color white Logo bottom of Detail & Forecast Screens (review requirements of using WU logo).

Investigate String Buffer use in speeding up concatenation in the XPP.

Set up Subversion from Home. Reimage kitchen box and set up as local cloud.

Add custom exception to KTime and replace exiting try/cache where called.

If Widgets exist on the desktop, do not stop service when exiting from the Slate.

When starting WT from the icon, always bring up the Slate. Confusing to bring up a random weather location if it was left at the top of the Task stack.

Put Web Services onto Azure via BizSpark account. Put latest version on phone for testing.

Widget Updates: Both Activities & Services DO NOT live for long periods of time (>24 hours). The OS kills them if they are not being used for a length of time. Need to use the AlarmManager to periodically call the PulseStatus service.

Detail Screen display redone to feature descriptions. Add Season to Details Screen: Summer, Winter, Spring, Fall. Provide Astrological and Meteorological options. Add Pressure Rate to Details Screen: Steady, Rising, Falling. Add Temp Average to Details Screen: Above, Below, Normal. Rearrange weather stats: wind / pressure first. Maybe put some line space in there.

Alerts: Create an Activity for the Alert Message and an Alert icon. Have the Alert Icon show up on Widgets and Slate.

Detail Screen: Better manage Condition text to fit on screen (example Heavy Snow Blowing Snow), e.g. do not insert "\n" before 12th character. Create a two level Above/Below Average, maybe "Warmer than Avg." More space under section 1.

Add real clouds to the cloud icons.

Allow for demo web download onto phones. Just generate an .APK file by running the application from Eclipse. They the phone can download the .APK file from anywhere.

Get rid of the WU icon on Slate search. Replace with Icon+Tunnel. Get rid of Screen Header too. Use the Action Bar Search Icon instead of the Search Button (http://developer.android.com/design/downloads/index.html). Switch to ActionBarSherlock to get ICS look & feel. http://jakewharton.com/actionbarsherlock-four-point-oh/

Create a disk based cache of the last full API call. Just 2 records plus the time and query codes. Each time a full API call is made, have it check the disk first and if an exact match that is less than 1 minute old, use it. This should speed switching between Detail/Forecast but still let them act independently.

Now that the target version is 14, any use of the network on the main thread will crash the application. Detail/Forecast Screen read should be Asynchronous and sport a Progress Bar. Also calls to Geocoding API and Adding a new item need to be off the main thread.

Add Action Bar to Detail/Forecasts/Alerts with back navigation. Make Target Version 14, Minimum Version 8.

Incorporate Ad Service into Search Screen. Maybe Detail screen. Advertising Bureau Standards http://www.iab.net. Cost Per Impression (CPM) like newspaper. Cost Per Action (CPC, CPA, etc.) requires some action by the user. eCPM is the CPC * clicks per 1000 impressions. Google Ad Network: Add Sense (publisher) and Add Words (advertiser).

Terms of Service dialog / text that comes up upon first use. Link to EULA on web. If they do not accept, exit app.

Build a marketing / help web site to show off the product.

Put PrePaq capability in place for later addition of actual data into system.

WT-API Security. The security implantation will be a simple registration where the application requests a unique id. That id must be used to call all other methods.

Move the WS* to use run the working role on web server and support scale by multiple servers. Break up weather and utility web services into two interfaces. Fix issue of new+dead overloading per minute call limits.

Switch Station to modern Storage Client, do add/update instead of Delete + Add. Add with retries. Use the continuation token when loading the data.

Put into Market Place. Update Docs: Interface, Appendix B, Add TOS and Privacy as appendix.

Get at the bottom of the problem when updating the Table Storage in NewStation by reporting the Table Storage error (stored in the inner exception).

Put Audit table (SQL) in place and log all requests. The SQL Server 5GB size is free. Put in place full audit add/read/del methods.

Bug Fixes

1. When no city is found on search, put up a help message about using less of the name
2. Display NA if forecast temperature is not there (looks like a bug in WUI data, last day never has low).

3. Truncate elevation to a whole number.
4. If Humidity is below 70 and it says it is raining, ignore the rain (go clear).
5. WU sends xml with the Entity “&deg;” but with no DTD, so the parse fails. (switch to XPP fixes it).
6. Process Celsius forecast text and display if Celsius setting.
7. Settings - Allow display of distance in Metric (centimeters, meters, kilometers). Affects Visibility, Precipitation, Wind Speed, Stations, Elevation, Wind Forecast.
8. Detail Forecast: Second forecast needs extra space (above) if first forecast is only 1 line and second is 2. Indirectly fixed by adding wind expectation.
9. If wind is zero (0) and gusts are >0, then make wind = gusts.
10. If the Climate Web Service throws an exception, it shows up as a HTTP Server Error 500, so check for it.
11. Do not try and update the station information when the network is not available.
12. The Condition “fog” is getting ignored because it is 3 characters long.
13. The Status Class should have a “isValid” method defaulted to false. Do not copy unless true.
14. Settings - Display pressure in inches or millibars. millibars * 0.02953(@32 f). There was a web site that indicated temperature affected things, but I found no supporting evidence and everyone uses this conversion.
15. When adding Contacts, people are usually not going to enter in their own country, so check that something is there before saving it as the “Real” location.
16. If low forecast temp data is N/A, and POP > 0%, use high temp to determine Rain/Snow text (and default to Rain). Even if API fixes this bug, safer this way.
17. If secondary station has wind while the primary is zero, use that secondary value. (This is actually how it works, as zero values for wind are not copied over existing values.)
18. Screen refresh should be every 10 seconds, then back off to 30 seconds, but seems to be always 30 seconds.
19. The NMBR_NA and DATA_NA should be better used such that the actual value can be changed and that the value is outside the range of a valid data (-9999 is proposed by WU). Check all uses to make sure not using value as real data.
20. Elevation is now reported in Meters (change to feet in Copy).
21. Problems with Antarctica: Public Station does not work, so if we do not find a private station, we should just put in location code (zmv) at zero distance. That will give us something.
22. When phone emulator has cleared data and no location, default data (death valley) does not work. It also seems to fail to pick up the actual location info. Change to another interesting location with a weather station.
23. Better demarcate the forecast text from the measurement expectations info.
24. Add expected amount of rain/snow to forecast, when POP is > 0%.
25. Clean up any Exceptions that are occurring.
26. Forecast data for Europe (maybe near midnight local) seems to not see most current report, but it is there on Detail Screen.
27. Slate screen jumps back to top of list during refresh, it should stay where it is.
28. Turn word Thunderstorm into two words to avoid overruns in 10 day Forecast.
29. “Loaded time zone name for …” takes too long. Need to change use of date formatter. **Error was red herring, it just started working after reboot. But switched to custom parser KTime for now since dates have been a pain.
30. Slate condition description should be truncated to 18 characters plus ellipse , e.g. thunderstorms and rain
31. Forecast Detail should be from the location’s local time perspective. Right now UTC time is getting used.
32. Call the 10 day forecast instead of the 7 day in web service. The processing of 10 days required changes because the day name overlaps.
33. Find all uses of the DB “.close()” method and make sure there is a try/catch around it. It can lock up the app.
34. Time zone of Alert is incorrect. Displaying as if UTC, not sure how it is stored.
35. Alerts for non-US give slightly different data, e.g. no epoch time, description and message same thing.
36. Role Recycle does not trigger a save of the cache. Any options? ** Needed to add an OnStop().
37. Forecast Text: Put a line break + word “Later” + for 2 text description of day which is usually the night forecast.
38. Get rid of text part of 10 day forecast. Let a tap bring up text dialog.
39. If PulseStatus ran only 40 seconds ago, do not rerun it again (debounce).
40. Add a version node to the API path, e.g. /Climate.svc/rest/v1/utility/log :: where v1.0 is the version.
41. Forecast condition text moved under Title, get rid of “Expectation” just have wind.
42. Widget Name change to Weather Tunnel.
43. Preload multiple locations on initial Start up (new database). San Francisco, Singapore, Mumbai, London
44. Check if (external) Intent exists before calling: Maps, Contacts
45. Widget text is not very readable on day (blue) images. Change to bright char color.

46. Setting to display time in 24 hour or 12 cycles.
47. Add Contact: Invalid address displayed, not selected, but put on anyway. If select->unselect, works. (see Doris)
48. Night not updated on Slate (w/ Slim call). Temp=64, Night=0 but full call got 64.5 and real sunrise/sunset. Could be because slim is generated once before it became night. 1) rounding in slim is being done by WS, but should be done on device, 2) Night flag can be stale, need to calculate real current time when comparing to sunset/sunrise.
49. Limit Slate to 25 items to manage data flow better.
50. Delete non-working Activity Views in Eclipse and recreate, then check in everything.
51. Save Status in Detail Activity, do not bother requesting it for Forecast.
52. Remove GPS coordinates, add Elevation to Details.
53. For re-sorting Slate, should we add a database field to support that NOW.
1. Change detail background/delimiter lines to clear night.
2. Alert icon on widget never turns off once it turns on.
3. Put word “Add” on search icon.
54. Put the Alert Icon on the Action Bar of the Current tab.
55. Location Provider code fails with latest emulators. Need to be more flexible with naming providers.
56. On Alerts, limit location name to 23 characters.
57. Upon first power up in AM (after night’s sleep), the Alarm for the data updates does not fire right away. Need to start it somehow.
58. If the distance to a weather station is not given, what should it be saved as? See CopyStatus.
59. If the closest PWS does not have a temperature when adding, try for a backup. Some weather stations are wind only. Example Alison. http://api.wunderground.com/api/49aa28b6d6039d9c/conditions/q/pws:MCQ080.xml
60. On restart/resume of Slate got a “cursor already closed” message. *Made note in database class on how to deal with ClimateDB/Database/Cursors.
61. Put a second Add on Options menu in case the search icon is not inviting enough.
62. Navigation: Widget->Detail->Home->WT. If the WT was not running already, this ends up at the Widget Detail and back goes back to the desktop. Intercept Back and always launch Slate.
63. Add xhdpi resource directories (96 dpi). Check on image size of action bar search for all sizes.
64. Turn off Baseball prepaq. Will use an API based look up for these types of things.
65. Make all Weather Tunnel references have ®
66. If widget has an alert, then you remove it, then you re-add it and it does not have an alert, seems to stay an alert for the call to details. Need to have details check if there is an alert before starting Alert Activity.
67. Geolookup seems to be outside of the normal counter and allows per second rate violations. Also, the total calls per day cap does not seem to work. ** Any lookup done after Daily Max reached will throw. To allow for Minute Max overruns, set the maximum a bit below the real number.
68. (Test locally with Alison data) On Slim Service, if any values are blank, reset them to defaults. Make Temp -9999.
69. Put Weather Underground logo on regular page/list, use bottom area for advertising.
70. Add location / keywords to AdMob. D:\Temp\hold\GoogleAdMobAdsSdkAndroid-6.0.1\docs\index.html. Center Adds in Landscape mode.
71. Make sure service/alarm starts upon widget Add, since may not have started main activity. TEST on phone.
72. Widget Nav has changed. If hit home from detail, clicking on a widget brings up the old detail. **Not sure what caused this, but uninstall/reinstall seemed to fix problem.
73. Allow selective call to retrieve status without Forecast. On WT, only call for forecast on one of the requests.
74. Current Location default should be Antarctica or a non-popular place like Tome NM. Otherwise a lot of people would not see “current location” upon first use.
75. Have a help link on About Screen pointing to wt_app.html.
76. Place try/catch on all data base methods in Slate.
77. Calculate wind based on both public/private stations. 50/50 adjusted for distance between near/far adjusted 5% per mile. E.g. (50 - Distance * 2.5) * .01 = % of wind speed public will contribute. This is only done when a private station is the primary.
78. Allow PWS: entry. Only use that data, not public station so they get only their data.
79. Add MAP to Details menu. Remove location link to map.
80. Move About from Menu item to Settings… and get rid of WU TOS.
81. Allow back to cancel Search/Load popup dialog.
82. Change alarms to use inexact option.
83. Add a GPS option to settings, default off. Do not use GPS unless it is on.

84. Move stats to Memcached: Stations* & Searches. New Stats: Add last persistent load timestamp for each role, Add Registration count stat, Add log record count for past week, Add a persistent non-dead station count, Previous 24 hour count plus time of switch over, current machine time.
85. Test on Low and Medium and High(button-less) resolutions in the device simulator. See Google recommended screen sizes. http://developer.android.com/guide/practices/screens_support.html
86. Getting errors when trying to store items in the Web Role. Reported as common when doing Deletes + Add. ***Switched to newer Add/Update method in Table Storage, added "with-retries".
87. Cannot put Toast calls in the non-UI thread of an AsyncTask. Fix to v1 of WT where there was a Toast call in the catch of the doInBackground() method.
88. When requesting the location by using a specific weather station, that weather station is not always the first returned. Need to just force it to be the one used (since we know that is what is desired).
89. Check on unhandled exception (database was inaccessible) in AsyncTask that caused crash on device (see Google Console). *** It is possible to get "database is locked" exceptions when getting a readable/writable database. While always using the same exact Helper is suppose to fix this, for now I am just sticking these in try/catch where I find issues. So far only 2 exceptions in 3 months.
90. If there is a failure in getting the weather station data from the WT-API, switch to the backup station. If the pws and public temp difference is greater than 50 and distance between is less than 13 miles, switch to the backup station.
91. The forecast does not show up immediately when switching from portrait to landscape. Put in code to allow tabs to stay in focus on change in device mode. Check if tab is in focus and if Status data is present before calling AsyncTask.
92. If device local time is future, looking at places in the past still says "next day". Change to "past day". *Added extra check for month change, in addition to day change.
93. If not tracking location, allow the first record to be hidden. If someone tries to delete first record, just hide it and turn off location tracking.
94. When the humidity is >= 95%, dew point is <= Normal and not rain – say there is fog.
95. Refresh on Detail display. This refresh just be the usual refresh, as if exiting and coming back into the Detail.
96. Improve copy on introduction screen. Use bullet points. Make dialog screen wider?
97. Widget Night/Day was confused with "haze" at night, put up a day time image. **Made Haze follow Fog for widgets. Still need better icons/graphics for these two conditions.
98. Add Humidity and Dew Point to "slim" WS method to support Fog.
99. Improve auto-start on boot: http://stackoverflow.com/questions/7978403/boot-receiver-not-work
100. Write tool to allow easy analysis of logs. Support date and text filter.
101. For some reason PUT is sometimes giving error that data is too large even though it is below the 64K limit. Need to be able to have a test harness that will look up a station from Table Storage to see if the station really did get added. ***After building log tool, this is not a very common occurrence and it probably an Azure bug. It also has not occurred in a while.
102. Improve graphics (icon/widget) for Fog and Haze.
103. Add record day high/low.
104. Update to latest OS/SDK on Azure (Server 2012 / SDK v2.0).
105. Need to save off log file and clear it out up to May 31.
106. Add web service retrieval to Log Tool (based on date range). Allow the data to be saved to disk. Allow data to be deleted from server. ***ViewLog.exe
107. The Delete dead items on the persistent load fails. May be a bug. Message is
108. 17201: : (- CoolftcWTClimate.LoadPersistentCache--WStationHelper.Delete(ByDate) -) Table Storage unable to process request. - InvalidInput?99:One of the request inputs is not valid. ** I think this may be due to multiple instances deleting from a single data store (cache).
109. The DbSqlServer class in Climatology should be refactored. Instead of a connection, just the connection string should be part of the constructor. Then each call should have a "using" create a database connection (and implicitly close it when the function exits. (see http://stackoverflow.com/questions/9055470/azure-sql-single-or-multiple-db-connections)
110. Log responses that are "not found" along with their details (see the notfound.xml). Note: It seems that this can be temporary.
111. Work on reducing specific errors or perhaps provide more specific details. Specifically: 1) Put in retry logic for SqlClient transport error, 2) Increased max allowed WU-API calls (must be some miss counting), 3) Only report WU server timeout in one place, 4) Increased the number of threads for external web service calls.

112. Replace WebClient with WebRequest, catch WebException with better handling.
113. If a station is requested and not found, do not add that station to the catch.
114. Have the Registration Audit records set to 9, and not deleted. Need simple call to clear out range of audit records to conserve space. Save off all but 90 days. Just use SQL Server Management Tool with scripts that delete a 15 days of data except for records with bill code = 9.
115. Fix second try an SQL call to Audit Table (must create new SqlParameter variable).
116. Add in 4 Dev WU accounts to boost available quotes. After hitting the exceed count, grab a random key based on the machine.
117. Remove logging of overflow accounts.
118. Add log message on retry for Audit to see if it was successful or not.
119. Add user tracking (from audit) to log messages. ***Logging supports this now, but the user information is not always available. It can be added later if deemed necessary.
120. Create a “not found” cache to intercept the most common query-not-found sites. Clear cache once a day. Does not have to be too precise, e.g. can be done with local (per box) memory.
121. Create a Ad-Free version
122. Upgrade to latest AdMob SDK.
123. Add WT version to GetRegistration and to ticket: Includes WS changes to store it in Audit table. Added “AndroidWTv<version>” to end of ticket.
124. Improve reporting using Count table. For registrations, just query SQL for recent ones to speed up. Do not bother with Station Cache size.
125. Improve the tracking of WU-API calls. Use the Count table directly based on midnight east coast.
126. Recode all Count based Stats to use the Count Database.
127. If the weather data is too big for the cache back store, just punt on that data. As long as it is a small percent of whole it should not be a problem. Table Storage is limited to 64KB, but for a Unicode string, that means only 32K characters. Most of the time the weather data is less than 32K characters, and would be convertible to non-Unicode except for the Alerts data (which requires Unicode). The Alerts data is also what pushes the character count up. In the future we might have to break Alerts data out if these records need to be kept in the backing store. On the bright side, Alerts only grow in size where there is severe conditions, so it should stay some reasonable percentage. Also, while we could just not store the Alerts part of the data, that seems like it would be a problem for users, if they missed the alert.
128. The QNF cache needs to be occasionally cleared out. It naturally seems to be dropped by the system at random, but it can grow pretty large. First try, lets just clear it out at records that are older than 1 day.
129. Add counts of Backing Store to stats page.
130. Fix Delete of backing store on Load to it no longer gets the “ContentLengthExceeded?The content length for the requested operation has exceeded the limit (4MB)” error.
131. The RUN is showing a “Object reference not set to an instance of an object.” Error. This was due to throwing a custom error inside an existing try catch that did not expect it. Just moved “throw” out of try catch block.
132. Add Count for “non response” to client in the API requests. Show it in the statistics.
133. Alerts can be just on Public, not on Private, so the slim will not see it but the detail will. But reading the warning, it excluded the Private area, so the display of the detail was incorrect – not the slim.
134. “Feels Like” Temp should only be shown if >1.5 degrees difference.
135. Allow alternate date formatting - http://en.wikipedia.org/wiki/Date_format_by_country . Settings: ListPreference. Choose the order of days/months/years in displayed dates. 02/21/2014 (MM/DD/YYYY), 21/02/2014 (DD/MM/YYYY), 2014/21/02 (YYYY/MM/DD). Also change the Long date format to be similar.
136. On the detail screen, add line break to current forecast if less than 55 characters.
137. Setting to show weather station ids.
138. Increase size of special.
139. Entering in a zip code with a leading zero does not put the zip code in Special.
140. Add in Google Analytics. Track # of users, screens.
141. Allow reorder Slate items using gestures. Have it as a menu item on action bar, and show a dotted vertical bar when enabled. Disable when the screen loses focus. Remove from settings, remove swipe delete.
142. Some contacts not getting metro (city) from google contact provider, don’t overwrite good data with this blank data.

Rejected

Current Location. Take the location logic out of the update loop in UpdateStatus and have it work in its own loop. **Using isForced when location changes is better idea. This ties up the db and is redundant.

Current Location. Periodically check for the location (every 30 minutes?), not just on resume, if the current location Widget exists. May be tricky without an Activity. ** See notes in QuickView.onUpdate(), but let's see if we can live with what we have. In general, battery use is an issue when doing stuff like location checks periodically forever.

Icon name change to W-Tunnel for better readability on desktop. **Live with it.

Navigation between Widget, Detail and Slate redesign. Google design guide recently addressed this issues. (https://plus.google.com/113735310430199015092/posts/ADZRqnt7PHj) The suggestion is to always return back up the "expected" stack. ** Note: This requires sometimes inserting the Slate in the back stack. The APIv11 supports "TaskStackBuilder" to assist with this. At this time the App is at v8, so have to allow for sporadic behavior. ** Another possibility might be to create a duplicate Detail Activity that eats the back and always sends the used back to the Widget. That is different than the recommendation but might be more consistent (since started the App clears the stack and always starts with Slate it is not an issue for that navigation).

Returning to Slate after HOME press away from Slate does not update location as fast as just coming in fresh. **This is intentional, as we trigger a delayed update in Resume so we can pick up any changes from stale data at the WS. Not that much of a difference.

Slate updated place & time, but not temp when there was a change of location. ** Unable to reproduce.

If no widgets, the Alarm is never started and the list gets old. In this case we should start the Alarm but use 1 hour when no widgets. ** The app will update pretty quickly when a person just goes in, so let's not start the Alarm.

Allow install to SD Card. ** This is easy to do with a manifest change, but will cause problems if they install to SD and then connect their phone to their computer. Loss of widgets & alarms. Can also never get the boot notification. App is not too large, so just saying no.

On the Nexus 4, the Alerts image shows up extra small. I could not see an easy way to get this to work. I think that phone tries to make a square out of the image and ends up shrinking it. Not an issue on any other phone I have come across, although it is a newer OS than any other I have seen. If it is an OS issue, could try using a different graphic for that OS.

Sometimes the Slate will freeze and not show updated data (this includes the time). Backing out will not stop it. Android will eventually kill it, so not too noticeable by users.

What is <estimated> Should we support it? No idea, but the one time I did see it it looked like this: <estimated> <estimated>1</estimated> <description>These are estimated conditions. There are no weather stations nearby.</description> </estimated> This was from pws:KNDGRAND8 around 12/30/2013.

Appendix A – WU API

Notes on the use of the Weather Underground API.

The API has two forms: Auto Complete and Full Data. The Auto Complete is used just for city lookup and does not require a key (is free). The Full Data is used to return all the Weather Underground data.

API keys: See InfoAccounts.docx.

Auto Complete:

http://autocomplete.wunderground.com/aq?query=<search>&format=xml

:: where <search> is a name or code in the WU database.

See the "API Autocomplet Results.txt" file for some query results.

The search term is primarily city names, although there are other things in the database of locations that are searched, e.g. zip codes, airport codes, countries, etc. You can limit by a CAPITALIZED country code if you know it. See file Country Codes ISO_3166-1_alpha-2.txt from- http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2. Search results include location name, type of location, country, link and City names (or other data stored as city name like zip codes or airport codes). At one time, you could drill down into countries & states for more cities, but this feature was (apparently) discontinued. WT filters out the countries & states in results.

The Lat/Lon & "zmw" as keys do not work, since just a simple text lookup.

Each city includes a link (zmw) that can be used to retrieve full location data. Alternatively, the city name could be parsed from the name and then preceded with the state/country code. The link for US cities is primary the zip code, plus an index. The search results are not always as correct as one would hope. One problem is that zip codes can cover two cities, which confuses things. This can lead to a zip code most people consider city X, showing up as city Y. You might even get an air port instead of a city. In a future version, getting the location information from Google and using lon/lat might be better.

It is possible to get a result in the search that will not return any data from the Full Data API. This is probably the database in need of some clean up. Or maybe it is used for other purposes (search China limited by &c=CN). Ultimately, to get the data on a specific location use the geolookup feature. It will return the same data as the original API. The geolookup can still give out multiple results for Country/City searches, so make sure it only gets the zmw link or lon/lat.

Full Data

http://api.wunderground.com/api/<key>/geolookup/q/<place>.xml

:: where <key> is the API key.

:: where < place> is the location. Location is typically one of two links: lon/lat(50.0564718,19.9512535) or zmw:link (zmw:94978.1.99999.xml). These two links can be found using the Auto Complete WU API, or the Google Geocoder API.

See the "API WUGeoCode Results Home.xml" file for a query result.

The Full Data lookup can return data that is valid at the time, but later is removed from the WU current conditions database. I assume that this is due to weather stations getting shut down or failing to report in a reasonable manner. On their web site, they have an aggressive algorithm to exclude weather sites as the primary data. Perhaps if a site stays unreliable for long enough its is removed from the current conditions altogether. Note that the station will still have history if it ever existed, but will result in a "query not found" when looking for current data.

---

http://api.wunderground.com/api/<key>/conditions/forecast7day/astronomy/q/<station>.xml

:: where <key> is the API key.

:: where <station> is the weather station. The stations are found as part of the geolookup. They come in two types: Public Weather Station (KDVO) and Personal Weather Station (pws: KCAFAIRF11).

See the “API Conditions Results Personal.xml“ or “API Conditions Results Public.xml” file for query results.

Appendix B – Graphics and Color

Colors used by the Weather Tunnel

Name	**Value**	**Description**
BasicRed	ff0000	Used for some text on the Alert screen.
BasicGreen	00ff00	Used for text on the Widgets
BasicBlue	0000ff	Not currently in use.
BasicBlack	000000	Not currently in use.
BasicWhite	ffffff	Not currently in use.
YellowBrick	ffd700	Used for “Special” text field.
BlueBrick	3360ff	Used for most of the Activity’s backgrounds.
BlueLiteBrick	3360ff	Used for banner background on Alert screen.
RedBrick	8C3D45	Used for the horizontal lines on some Activities.
GreenBrick	4abb4a	Not currently in use.
BrightChar	ccffff	Generally used as the color for text on colored backgrounds.
BrightRed	ff4040	Not currently in use.
OrangeBrick	ff8040	Not currently in use.
ClearYellow	d6d61d	Not currently in use.
ClearGreen	51de00	Not currently in use.
ClearNight	3c3c5e	Used as a dark color for text instead of just black.
BrightYellow	ffff00	Used for some text.
BayBridgeWhite	f5f5f5	Used as the Activity background for Alerts.

Graphic size suggestions:

Item	**Dimensions**	**Description**
ldpi	36 x 36	Low resolution folder.
mdpi	48 x 48	Medium resolution folder.
hdpi	72 x 72	High resolution folder
Warning	½ normal: 18, 24 and 36 pixels	The warning symbol used for alerts is ½ the normal dimension for the resolution folder it is in. This means it will take up ½ the space on the screen.
WarningSm	18 x 18	The warning symbol, used on widgets and the slate. Same size as the Warning stored in the ldpi folder.
left, right, up, down	¾ normal: 27, 36, 54	The directional images that indicate a person can tap on them and proceed. Basically a colored triangle. Even though the canvas is this size, the actual image does not take up the full space. This makes it visually appropriate (small) but still provides reasonable space for a finger to hit it.

Appendix C – QA

Client

Run all screens and widgets on multiple Screen Sizes / Densities. For each screen size perform the following:

1) Start out with the "wiped data" option to get a fresh device. The default cities are fine to work with.
2) Verify TOS screen readable.
3) Set GPS Coordinates in Eclipse to home: 37.998564 -122.570269
4) Find a city with a weather alert and add it. http://www.wunderground.com/severe.asp
5) Add 4 new cities: Johan, New York, Mexico, Hong Kong… should work. (use Add Place button & Search Icon)
6) Add a person to contacts (outside of WT) Alison Conor @145 S. Holliston Pasadena CA 91106. Return to WT and add a Friend.
7) Delete one of the cities from the list. Not the one with an Alert.
8) Go into Details for a non-Alert city. Verify spacing and measurements. Check on Forecast. Tap on one forecast entry to see popup.
9) Go into Details for an Alert city. Tap on the Alert. Verify Alert is readable. If more than one alert, check all.
10) Go to Settings. Check that Help / About screens work. Change to European Mode.
11) Go into details for a city, check that European mode working.
12) Add a few widgets. Use Alert city, Night City, different conditions. Switch to Landscape mode while they are on the screen. Tap on a widget to bring up WT.

Server

Appendix D – Production Installs

https://play.google.com/store/apps/details?id=coolftc.android.weather.tunnel

http://www.coolftc.org

Android (in eclipse):

1) Set DEBUG_RUN = false in ExpClass
2) Add 1 to the versionCode and adjust the versionName in Manifest (see below)
3) Update the "ver_" information in the string resources (for the About screen) such that the x.y.z.a has x.y equal to the versionName. The z.a can be used for one-off or test builds.
4) Delete everything in this path: K:\Applications\Projects\Weather Tunnel\Source Code\Android\coolftc.android.weather.tunnel\bin\dexedLibs
5) Generate signed .apk with correct key. Right click on project, Android Tools -> Export Signed Application Package. Use <developer.key.coolftc.android> in External directory. Use same password for both file and alias. Use full name for file: coolftc.android.weather.tunnel.apk.

One Off Builds:

SD Card friendly .apk. To get this to work, change android:installLocation="auto" in the manifest (from internalOnly).

Azure:

1) Set Debug Mode = false in Web Configuration.
2) Verify using production WU account.
3) Verify using production data storage for Cloud.
4) Package WeatherTunnelAPI (will generate files) if want to do it old way.
5) Publish to Stage. I swap the deployment label between A and B. Test.
6) Swap IPs.

*Per Google:

versionCode An integer value that represents the version of the application code, relative to other versions.

The value is an integer so that other applications can programmatically evaluate it, for example to check an upgrade or downgrade relationship. You can set the value to any integer you want, however you should make sure that each successive release of your application uses a greater value. The system does not enforce this behavior, but increasing the value with successive releases is normative.

Typically, you would release the first version of your application with versionCode set to 1, then monotonically increase the value with each release, regardless whether the release constitutes a major or minor release. This means that the **versionCode** value does not necessarily have a strong resemblance to the application release version that is visible to the user (see **versionName**, below). Applications and publishing services should not display this version value to users.

versionName A string value that represents the release version of the application code, as it should be shown to users.

The value is a string so that you can describe the application version as a <major>.<minor>.<point> string, or as any other type of absolute or relative version identifier.

As with **versionCode**, the system does not use this value for any internal purpose, other than to enable applications to display it to users. Publishing services may also extract the **versionName** value for display to users.

Problem: On Export got an Error 1. Make sure the android support library only exists in ABS (all references must point to that one file). Then deleted everything in this path: K:\Applications\Projects\Weather Tunnel\Source Code\Android\coolftc.android.weather.tunnel\bin\dexedLibs.

Appendix E – Audit & Analytics

Audit

```
CREATE TABLE [dbo].[Audit](
        [AuditID] [int] IDENTITY(1,1) NOT NULL,          // Key
        [UserTrack] [varchar](255) NULL,                 // This is the user, might be another program
        [Machine] [varchar](50) NULL,                    // Identifies the instance executed on
        [ApplicationName] [varchar](50) NULL,            // The high level web service or role
        [MethodName] [varchar](50) NULL,                 // Method called
        [Parameters] [varchar](1024) NULL,               // Input to Method (within limits)
        [ParametersRaw] [image] NULL,                    // If regular Parameters will not work
        [Source] [varchar](250) NULL,                    // Usually IP address + App version
        [Billing] [int] NULL CONSTRAINT [DF_Audit_Billing] DEFAULT (0),        // Billing code
        [CreateDate] [datetime] NULL CONSTRAINT [DF_Audit_CreateDate]  DEFAULT (getdate()),
        [Signature] [image] NULL
```

The Audit table keeps track of what is going on in the system. It can grow pretty quickly, so it requires some normal cleanup over time.

The UserTrack will typically store the registration GUID for the user. For Support WS calls, it stores the special security key. For the role instance it just has the word AsyncUpdater.

The Billing value can store a number that can be modified without invalidating the signature, as it is not part of that calculation. This allows processing/workflow of the record while keeping the main data untouched.

The Signature is an encrypted hash of all data (except billing code) in the record. This provides a verifiable record that the data has not be tamper with.

The Billing Codes

0	This is the default billing code. This is also the ST_CACHE.ST_READY enum value.
0 – 4	Possible values returned when checking the cache for weather data. These values are defined in the ST_CACHE enum, explained below.
9	Calls to GetRegistration use this value. The GetResgistration records are interesting and will be stored for a longer time, i.e. they are not part of the normal cleanup.

For methods that get weather data, the return value from a check of the cache is stored as the billing code using this enum.

```
ST_CACHE {
        ST_READY,           // The data is young enough to return.
        ST_STALE,           // The data in the cache should be updated, but the data can be returned.
        ST_DEAD,            // The age of the data in the cache is too old to use, get new data.
        ST_NOTFOUND,        // Nothing matching in the cache, get new data.
        ST_WEBROLE          // Updating the cache from the stale queue, using the Web Role thread[1].
}
```

[1] The Web Role code does not have access to this enum, so it just uses the constant 4.

Maintenance

The Audit table is stored on a SQL Server instance known as ox7kxu91l1.database.windows.net, with a Database called WTunnel, in a table named Audit. Since every call to the Web Service places at least 1 record in the Audit table, it can grow fast and large. The plan as of now is to keep the database to 10MM records. As it approaches the desired number of records, a trailing month of data will be removed from the database, except for GetRegistration records. These records, with a billing code of 9, are interesting and not common.

The SQL script to clean up the data (one month at a time) can be found in the Test url List.txt file at K:\Applications\Projects\Weather Tunnel\Test QA.

Analytics

```
CREATE TABLE [dbo].[Count](
        [CountID] [bigint] IDENTITY(1,1) NOT NULL,      // Unique value part of key
        [Counter] [int] NOT NULL,                       // The type of counter event
        [CreateDate] [datetimeoffset] NOT NULL CONSTRAINT [DF_Count_CreateDate]  DEFAULT (getutcdate())
 CONSTRAINT [PK_Count] PRIMARY KEY CLUSTERED
( [CountID] ASC, [Counter] ASC, [CreateDate] ASC ) WITH (IGNORE_DUP_KEY = OFF))
```

The Count table provides a storage area for events that need to be counted in the application. Anytime a specific event occurs, a record with a specific type (Counter) value is written to the table. This allows later counting of specific events, with a conditional value of time. Most analytics are based on counting the number of something occurring over a span of time. This type of table is very efficient because Adds are quick, and Adds are most of what is going on. The queries are efficient too, as they are always counting the number of rows, with the only conditions being part of the key.

The Count Types are as follows (check the code for the latest):

```
public enum CNT_CODES{
        CNT_WUAPI_CALLS = 1,        // Weather Underground API tries
        CNT_WUAPI_WORKS = 2,        // Weather Underground API success
        CNT_CACHE_QNF = 3,          // Query Not Found cache hit
        CNT_WUAPI_QNF = 4,          // Query Not Found rejection on API call
        CNT_WT_LOG = 5,             // An entry was written to the log
        CNT_WT_SEARCH = 6,          // Each time a Search is called
        CNT_WT_WSBASIC = 7,         // Weather Station Basic
        CNT_WT_WSFORCE = 8,         // Weather Station Forced
        CNT_WT_WSSMALL = 9,         // Weather Station Small/Slim
        CNT_WT_WSDETAIL = 10,       // Weather Station Details
        CNT_WT_PREPAQ = 11,         // Get a PrePaq
        CNT_WT_SIGNUP = 12,         // Registration
        CNT_WT_QUEUE = 13,          // Processed out of weather queue
        CNT_CACHE_LOAD = 14         // Count of stations loaded into the cache from storage.
};
```

Appendix F – SDKs

Weather Tunnel uses a number of SDKs and components.(**Removed in later versions**)

AdMob

The AdMob SDK provides AdView, suitable for showing banner ads in many different sizes and InterstitialAdm, which provides a means to display interstitial ads (image and video). The App uses the v6.4.1 of Google Mobile Ads. Google is pushing a "Google Play" based ad service, but it does not work with Android v2.2.

Docs: https://developers.google.com/mobile-ads-sdk/

Note: In the App you can set ADS_HIDDEN = true to hide the ads, for screen shots or picky users.

To change to a newer SDK, go to Properties -> Java Build Path and add in the new JAR file (and remove the old).

Appendix G – Legal Docs

Privacy Statement

Weather Tunnel, Developed by zalicon enterprises, inc.

This privacy statement applies to the Weather Tunnel Android Application (the Application) developed by zalicon enterprises, inc. (the Developer), excluding third party Advertising components used to show display ads in the Application. Ad Service components operate outside the control of the Weather Tunnel and abide by their own policies.

A comfortable experience with the Application is our desire. To that end the Application does not use the information it collects outside the Application, and all collected data is done so to support specific features of the Application.

Information Collected

The Application, on each phone, collects a unique id (generated by the OS) and stores it on a server to prevent abuse of the Web Services crucial to the operation of the Application. No personal identifying data is stored external to the device by the Application.

The Application, in the course of its operation, gathers information about locations. Locations may include the current location of the device, the location of people's addresses located in the device Contact storage and locations a user may randomly search and save. All this data is stored locally on the device.

Information Use

The Application and its Developer do not use data collected for any use other than enabling the functionality of the Application. Data is not collected with any goal other than supplying an enjoyable experience in viewing the weather. The Application does not ship information (other than that mentioned above) outside the device.

The Application may in the future provide information to In-App Ad Service components, if that is found to improve their performance. Although no personal indentifying data will ever be used in that way.

Avoiding features that use Personal Data

To avoid having the current location tracked, the Application's Setting's screen allows a user to stop the application from tracking the current location of the device. The use of Contacts only occurs by user initiation, via the Add Friends menu option. Do not use that option and the Contacts will not be accessed. Upon uninstall of the Application, the local database is deleted and is not stored in any other location by the Application.

NOTE: Third party Ad services used for display ads are out of the control of the Application and Developer. They may or may not be using information available on the device.

Currently the following Ad services are used (with links to their privacy polices):

AdMob: http://www.admob.com/home/privacy

End User License Agreement

PLEASE CAREFULLY READ ALL SECTIONS OF THE FOLLOWING END USER LICENSE AGREEMENT (AGREEMENT), WHICH APPLIES TO THE WEATHER TUNNEL® MOBILE SOFTWARE APPLICATION (SOFTWARE). BY DOWNLOADING OR USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL TERMS OF THIS AGREEMENT. IF YOU DO NOT WISH TO ACCEPT ALL OF THE TERMS OF THIS AGREEMENT, DO NOT DOWNLOAD OR USE THE SOFTWARE.

LICENSE.
----
zalicon enterprises, inc. (zalicon) hereby gives you a personal, non-exclusive license to use the software application Weather Tunnel®. This license is granted for as long as the Software is supported by zalicon (Term). Any new versions of the Software are also covered by the Agreement.

You may:

- install the Software an unlimited number of times on any supported devices that you control;
- use the Software for your personal, non-commercial use only;

You may not:

- run the Software on devices that circumvent in-application advertising;
- modify, translate, reverse engineer, decompile, disassemble (except to the extent applicable laws specifically prohibit such restriction) the Software;
- create derivative works based on the Software;
- copy the Software;
- rent, lease, transfer or otherwise transfer rights to the Software.

Your use of the Software is strictly your choice. The software can be uninstalled at any time.

TERMINATION AND MODIFICATION.
----
zalicon may, in its sole discretion, terminate the Agreement without cause at any time. Your license to the Software and zalicon's obligations under this Agreement will automatically terminate if you fail to comply with any term of this Agreement. The indemnification obligations contained in the INDEMNIFICATION sections shall survive termination of this Agreement for one (1) year.

WARRANTIES.
----
zalicon warrants to you during the Term that the Software will achieve in all material respects the functionality described in the User Guides applicable to the products acquired by Customer and that such functionality will be maintained in all material respects in subsequent upgrades to the Software. zalicon does not warrant that the Software will be error-free. Customer's sole and exclusive remedy for zalicon's breach of this warranty shall be that zalicon shall be required to use commercially reasonable efforts to modify the Software to achieve in all material respects the functionality described in the User Guides and other related documentation and if zalicon is unable to restore such functionality, Customer shall be entitled to terminate the Agreement and shall be entitled to receive a pro-rata refund of any license fees paid for its use of the Software for the terminated portion of the Term. zalicon shall have no obligation with respect to a warranty claim unless notified of such claim within sixty (60) days of the first instance of any material functionality problem, and such notice must be sent to software.support@zalicon.com.

zalicon warrants that it is the sole owner and has full power and authority to grant the license and use of the Software and other rights granted by the Agreement to you with respect to the Software and that neither the performance by you in its utilization of the Software, nor the license of and authorized use by you of the Software as described herein will in any way constitute an infringement or other violation of any copyright or trademark of any third party.

zalicon warrants that the Software shall be free of viruses, Trojan horses, worms, spyware, or other malicious code or components.

ZALICON DOES NOT REPRESENT THAT YOUR USE OF THE SOFTWARE WILL BE SECURE, TIMELY, UNINTERRUPTED OR ERROR-FREE OR THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS OR THAT ALL ERRORS IN THE SOFTWARE AND/OR DOCUMENTATION WILL BE CORRECTED OR THAT THE SYSTEM THAT MAKES THE SOFTWARE AVAILABLE WILL BE FREE OF VIRUSES OR OTHER HARMFUL COMPONENTS. THE WARRANTIES STATED ABOVE ARE THE SOLE AND EXCLUSIVE WARRANTIES OFFERED BY ZALICON. THERE ARE NO OTHER WARRANTIES OR CONDITIONS, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THOSE OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. EXCEPT AS STATED ABOVE, THE SOFTWARE IS PROVIDED TO YOU ON AN "AS IS" AND "AS AVAILABLE" BASIS. YOU ASSUMES ALL RESPONSIBILITY FOR DETERMINING WHETHER THE SOFTWARE OR THE INFORMATION GENERATED THEREBY IS ACCURATE OR SUFFICIENT FOR YOUR PURPOSES.

LIMITATIONS OF LIABILITY.

YOU AGREES THAT THE CONSIDERATION WHICH ZALICON MAY CHARGE HEREUNDER DOES NOT INCLUDE CONSIDERATION FOR ASSUMPTION BY ZALICON OF THE RISK OF YOUR INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT SHALL EITHER PARTY BE LIABLE TO ANYONE FOR INCIDENTAL, CONSEQUENTIAL,

PUNITIVE, SPECIAL OR EXEMPLARY DAMAGES, OR INDIRECT DAMAGES OF ANY TYPE OR KIND (INCLUDING, BUT NOT LIMITED TO, LOST REVENUE, LOST PROFITS, OR LOSS OF OTHER ECONOMIC ADVANTAGE) ARISING FROM BREACH OF WARRANTY, BREACH OF CONTRACT, NEGLIGENCE, OR ANY OTHER LEGAL CAUSE OF ACTION TO THE MAXIMUM EXTENT PERMITTED BY LAW ARISING FROM OR IN CONNECTION WITH THIS AGREEMENT. Except with regard to amounts due under this Agreement, the maximum liability of either party to any person, firm or corporation whatsoever arising out of or in the connection with any license, use or other employment of the Software, whether such liability arises from any claim based on breach or repudiation of contract, breach of warranty, negligence, tort, or otherwise, shall in no case exceed the equivalent of original price of the Software. The essential purpose of this provision is to limit the potential liability of the parties arising from this Agreement. The parties acknowledge that the limitations set forth in this Section are integral to the amount of fees charged in connection with the license of the Software and that, were zalicon to assume any further liability other than as set forth herein, such fees would of necessity be set substantially higher. Certain states and/or jurisdictions do not allow the exclusion of implied warranties or limitations of liability for incidental or consequential damages, so the exclusions set forth above may not apply to Customer. THE LIMITATION OF LIABILITY SET FORTH IN THIS SECTION SHALL NOT APPLY TO EITHER PARTY'S INDEMNITY OBLIGATIONS SET FORTH IN THE INDEMNIFICATION SECTION BELOW.

INDEMNIFICATION.
----
zalicon will indemnify, defend and hold you harmless from and against any and all costs, liabilities, losses, and expenses (including, but not limited to, reasonable attorneys' fees) (collectively, "Losses") arising out of or in connection with a claim, suit, action, or proceeding brought by any third party against you which arise out of or result from the infringement of any copyright, trademark, or misappropriation of a trade secret relating to the Software; provided that you (a) promptly give zalicon notice of the claim, suit, action, or proceeding; (b) give zalicon sole control of the defense and related settlement negotiations; and (c) provide zalicon with all reasonably available information and assistance necessary to perform zalicon's obligations under this paragraph. If the Software is held to infringe, zalicon will, at its own expense, in its sole discretion use commercially reasonable efforts either (a) to procure a license that will protect you against such claim without cost to Customer; or (b) to replace the Software with a non-infringing Software. Provided that zalicon complies with this Section, you shall be entitled as its sole and exclusive remedy to terminate the Agreement.

MISCELLANEOUS.

If any provision or provisions hereof shall be held to be invalid, illegal, or unenforceable, the validity, legality, and enforceability of the remaining provisions shall not be in any way affected or impaired.